

68

MICRO JOURNAL

Australia A \$7.95 New Zealand NZ \$9.85
 Singapore S \$14.95 Hong Kong H \$29.00
 Malaysia M \$14.25 Sweden 30:-SEK

\$2.95_{USA}

OS-9 Atari Amiga Mac S-50

68000 68010 68008 68000 68010 68020 68030

The Magazine for Motorola CPU Devices For Over a Decade!

This Issue:

Basically OS9 p. 13

"C" User Notes p. 6

68XXX ROM Monitor p. 24

MUMPS - PC68K1 - OS9 p. 20

Logically Speaking p. 38

The Hit Bucket - Motorola News Releases, Letters, Updates etc.

OS-9

SK-DOS Atari Amiga

FLEX Macintosh

A User Contributor Journal

And Lots More!

VOLUME XI ISSUE V • Devoted to the 68XXX User • May 1989

The Grandfather of "DeskTop Publishing™"

SERVING THE 68XXX USER WORLDWIDE



PHOTO: GREGG MAG



74470 12810

WHO DO YOU CALL WHEN YOUR DEBUGGER WON'T DEBUG?



The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

Why Not Try the Microware One-Stop Total Solution?

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT&T.
VAX is a trademark of DEC.

Modularity Lets YOU Choose Just What You Need.

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

Support is Part of the Package.

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

The OS-9 Success Kit

A Total Integrated Solution for Your Next Project

Development Tools:

C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

Kernel Options:

MMU (Security Protection) Support
Math Coprocessor Support

* Resident or UNIX versions available
** VAX hosted

Languages:

C*
Basic
Pascal
Fortran
Ada**
Assembler*

I/O Options:

SGSI, SASI & SMD Disks
3-, 5-, 8-inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

microware® OS-9

Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

Microware Japan Ltd.
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

Mustang-020 Mustang-08 Benchmarks

IBM AT 7300 Xmas Sys 3
AT&T 7300 UNIX PC 68010
DEC VAX 11/780 UNIX Berkeley 4.2
DEC VAX 11/750
68000 OS-9 68K 8 Mhz
68000 OS-9 68K 10 Mhz
MUSTANG-08 68000 OS-9 68K 10 Mhz
MUSTANG-020 68020 OS-9 68K 16 Mhz
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz

| 32 bit longer | Register Long |
|------------------|------------------|
| 9.7 | |
| 7.2 | 4.3 |
| 3.6 | 3.2 |
| 5.1 | 3.2 |
| 18.0 | 9.0 |
| 6.5 | 4.0 |
| 9.8 | 6.3 |
| 2.2 | 0.88 |
| 1.8 | 1.22 |

Main()

register long i;
(for (i=0; i < 999999; ++i);

Estimated MIPS - MUSTANG-020 4.5 MIPS.

Burn to 8 - 10 MIPS: Motorola Specs

OS-9

| | |
|---|----------|
| OS-9 Professional Ver | \$850.00 |
| *Includes C Compiler | |
| Basic09 | 450.00 |
| C Compiler | 500.00 |
| 68000 Disassembler (w/source add: \$100.00) | 100.00 |
| Fortran 77 | 750.00 |
| Microware Pascal | 500.00 |
| Oneseg09 Pascal | 900.00 |
| Style-Graph | 495.00 |
| Style-Spell | 195.00 |
| Style-Merge | 175.00 |
| Style-Graph-Spell-Merge | 695.00 |
| PAT w/C source | 229.00 |
| JUST w/C source | 79.95 |
| PAT/JUST Combo | 249.50 |
| Scalpac+ (see below) | 995.00 |
| COM | 125.00 |

UniFLEX

| | |
|----------------------|----------|
| UniFLEX (68020 ver) | \$450.00 |
| Source Editor | 150.00 |
| Scr-Merge | 200.00 |
| BASIC/Prof compiler | 300.00 |
| C Compiler | 350.00 |
| COBOL | 750.00 |
| MODEM w/source | 100.00 |
| MODEM w/source | 100.00 |
| X-TALK (see Ad) | 99.95 |
| Cross Assembler | 50.00 |
| Fortran 77 | 450.00 |
| Scalpac+ (see below) | 995.00 |

Standard MUSTANG-020TM shipped 12.5 Mhz.

Add for 16.6 Mhz 68020 375.00

Add for 16.6 Mhz 68881 375.00

Add for 20 Mhz 68020/RAM 750.00

16 Port exp. RS-232 335.00

Requires 1 or 2 Adapter Cards below RS232 Adapter 165.00

Each card supports 4 additional ser. ports
(total of 36 serial ports supported)

60 line Parallel I/O card 398.00

Uses 3 68230 Interrupt/Timer chips,
6 groups of 8 lines each, separate buffer
direction control for each group.

Prototype Board 75.00

uses for both dip and POA devices & a
pre-wired memory area up to 512K ORAM.

SBC-AN 475.00

Interface between the system and
ARCNET confined token-passing LAN. Fiber optics optional - call.
LAN software drivers 120.00

Expansion for Motorola I/O Channel Modules \$195.00

Special for complete MUSTANG-020TM system buyers - Scalpac+
\$695.00. SAVE \$300.00

Software Discounts

All MUSTANG-020TM system and board buyers are entitled to
discounts on all listed software 10-70% depending on amt. Call or
write for quote. Discounts apply after the sale as well.

Note: Only Professional OS-9 Now Available
(68020 Version) Includes (\$500) C Compiler -
68020 & 68881 Supported - For UPGRADES
Write or Call for Professional OS-9 Upgrade Kit

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP R M (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability



Don't be misled!
ONLY Data-Comp
delivers the Super
MUSTANG-020

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission,
Government Agencies as well as Universities, Business, Labs, and other Critical Applications
Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level
V compatibility and low cost is a must.

The
P
R
O
!

Only the "PRO" Version
of OS-9 Supported!



This is **HEAVY DUTY**
Country!

For a limited time we will offer a \$400 trade-in on your
old 68XXX SBC. Must be working properly and
complete with all software, cables and documentation.
Call for more information

Price List:

| | |
|---|---------------|
| Mustang-020 SBC | \$2490.00 |
| Cabinet w/winch PS | \$299.95 |
| 5"-80 track floppy DS/DD | \$269.95 |
| Floppy Cable | \$39.95 |
| OS-9 68K Professional Version | \$850.00 |
| C Compiler (\$500 Value) | N/C |
| Winchester Cable | \$39.95 |
| Winchester Drive 25 Mbyte | \$895.00 |
| Hard Disk Controller | \$395.00 |
| Shipping USA UPS | \$20.00 |
| UniFLEX | Less \$100.00 |
| MC68881 1/2 math processor | Add \$275.00 |
| 16.67 Mhz MC68020 | \$375.00 |
| 16.67 Mhz MC68881 | \$375.00 |
| 20 Mhz MC68020 Sys | \$750.00 |
| Note all 68881 chips work with 20 Mhz Sys | |
| Total: | \$5299.80 |

NEW LOWER PRICES
25 Mbyte HD ~~\$4299.80~~ \$3749.80
85 Mbyte HD ~~\$5748.80~~ \$4548.80

Data-Comp Division



A Decade of Quality Service

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 • Telex 510 600-6630 Hixson, TN 37343
FAX (615)842-7990

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Cheryl Hodge

Contributing & Associate Editors

| | |
|--------------|-----------------------|
| Ron Anderson | Dr. E.M. "Bud" Pass |
| Ron Voigts | Art Weller |
| Doug Lurie | Dr. Theo Elbert |
| Ed Law | & Hundreds More of Us |

Contents

| | | |
|--------------------|----|-----------|
| "C" User Notes | 6 | Pass |
| Basically OS9 | 13 | Voigts |
| Come On In | | |
| The Waters Fine | 20 | Plus Five |
| 68XXX ROM Monitor | 24 | Lunt |
| Logically Speaking | 38 | Jones |
| Bit Bucket | 46 | All Of Us |
| Classifieds | 58 | |

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"



68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 Telex 510 600-6630

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continuously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN, and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year \$24.50 USA, Canada & Mexico \$34.00 a year.

Others add \$12.00 a year surface, \$48.00 a year Airmail, USA funds. 2 years \$42.50, 3 years \$64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of \$15.50 for first 15 words. Add \$.60 per word thereafter. No classifieds accepted by telephone.

PAT - JUST

PAT WITH 'C' Source \$229.00

JUST WITH 'C' Source \$79.95

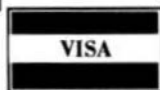
OS-9 68K - 68008 - 68000 - 68010 - 68020 - OS-9 68K

PAT FROM S. E. MEDIA — A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU!** All **PIE** features & **much more!** Too many features to list. And if you don't like ours, change or add your own. C source included. Easily configures to your CRT terminal, with special configuration section. No sweat!

COMBO **PAT**

JUST **Special \$249.00**

JUST from S. E. MEDIA — Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set -up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The **ONLY** stand alone text processor for the 68XXX OS-9, that we have seen. And at a very **LOW PRICE!** Order from: S. E. MEDIA - see catalog this issue.



Southeast East Media

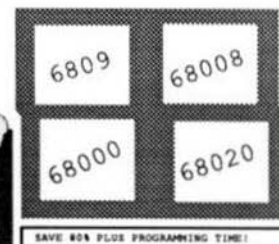
5900 Cassandra Smith Rd
Hixson, Tn. 37343
(615) 842-4600 FAX (615) 842-7990

Shipping

| | |
|----------------|----------------|
| U.S.A. | \$4.50 |
| CANADA | \$7.50 |
| FOREIGN | \$25.00 |

CLOSE OUT SPECIAL

SCULPTOR



**From the world's oldest
& largest OS-9 software house!**

CUTS PROGRAMMING TIME UP TO 80%
6809/68000-68030 Save 90%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develop DBMS type systems running under OS-9. A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.14:6

6809 - \$1295.00

68000 \$1295.00

68020 \$1990.00

**Due to a "Special One Time" Purchase, We Are
Making This Savings Offer. Quantities Limited!
*Once this supply is gone the price goes back up!***

System OS-9: 6809/68000-68030

• Regular ~~\$1295.00~~

ONLY

\$99.95

S.E. MEDIA

POB 849 5900 CASSANDRA SMITH ROAD
HIXSON, TN 37343 615 842-4601
TELEX 510 600-6630 FAX (615)842-7990



SAVE - WHILE SUPPLIES LAST!

Telephone: (615) 842-4600

South East Media
OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS-DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large main and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and - without any alterations to the programs - run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australia, the Americas and Europe - Sculptor is already at work in over 20 countries.

THE PACKAGE

With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run time system is available at a nominal cost.

Facts
■■■■■■■■

Features
■■■■■■■■

DATA DICTIONARY

Each file may have one or more record types. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Packed, fixed length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

- Unary minus
- * Multiplication
- / Division
- % Remainder
- + Addition
- Subtraction

MAXIMA AND MINIMA

- Minimum key length 1 byte
- Maximum key length 160 bytes
- Minimum record length 3 bytes
- Maximum record length 32767 bytes
- Maximum fields per record 32767
- Maximum records per file 16 million
- Maximum files per program 16
- Maximum open files 16

Operating system limit

PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen form program
- ☐ Generate standard report program
- ☐ Compile screen form program
- ☐ Compile report program
- ☐ Screen form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

RELATIONAL OPERATORS

- = Equal to
- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- <> Not equal to
- and Logical and
- or Logical or
- contains Contains
- begin with Begins with

SPECIAL FEATURES

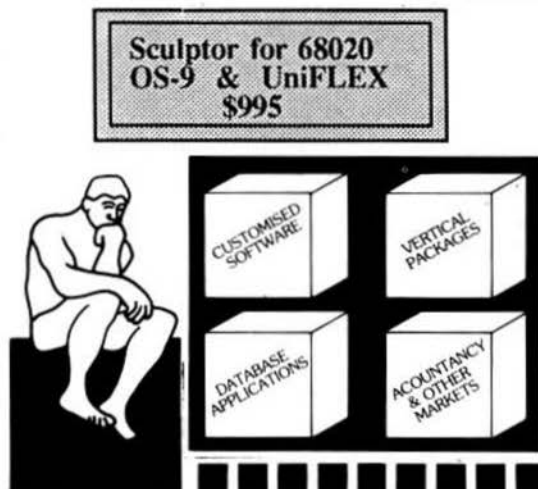
- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

QUERY FACILITY

- ☐ Query facility
- ☐ RefORMAT file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type



MUSTANG-020 Users - Ask For Your Special Discount!

MUSTANG-020

***\$1,990 \$398 \$795**

PC/XT/AT/MSDOS \$695 \$139 \$299

MUSTANG-08

***\$1,295 \$259 \$495**

Call or write for prices on the following systems.

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX, REGAL), STRIDE, ALTOS, APRICOT, ARETE, ARM-STRONG, BLEASDALE, CHARLES RIVERS, CMX, CONVERG.TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N.STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6809/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

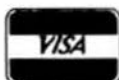
... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

- Full Development Package
- Run Time Only
- C Key File Library

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCB = Color Computer OS-9
CCF = Color Computer FLEX



South East Media

5900 Cassandra Smith Rd. - Hixson, TN. 37343
Telephone: (615) 842-4600 Telex: 5106006630



•• Shipping ••
Add 2% U.S.A. (min. \$2.99)
Foreign Surface Add 3%
Foreign Air Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola.*FLEX and UniFLEX are Trademarks of Technical Systems Consultants.*SK-DOS is a Trademark of Star-K Software Systems Corp.

C

*The C Programmers
Reference Source.
Always Right On Target!*

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter provides information about MINIX, a new operating system, similar to UNIX, which is now available for 680x0 processors.

It discusses several approaches to enhancing the use and debugging of C code containing memory allocation and deallocation.

It provides a new approach to program tracing under UNIX which is a considerable improvement over prior techniques.

It provides and solves a little problem of computing the next smaller power of 2 than a given number in the most efficient manner.

It presents a technique of using function arrays for efficiently switching control to one of several functions.

MINIX

MINIX is a new operating system that is system-call compatible with V7 UNIX. Unlike AT&T UNIX, it is available with all the source code, both the kernel and all the utilities. MINIX has been running on the IBM PC, XT, AT, and many clones, including 386s, for over two years. The IBM version is in widespread use worldwide and was written by Andrew S. Tanenbaum.

It has recently become available on a 68000-based computer, the Atari ST. When MINIX is run on the Atari ST (or MegaST) it replaces the native operating system (TOS) and turns the computer into a normal UNIX machine. The Atari port was done by Johan Stevenson and Jost Muller.

Several groups are currently attempting to port it to other 680x0-based computers (Amiga, Macintosh, PT, Terminus, etc.).

MINIX does not contain any proprietary AT&T code. Thus, both the operating system and the utilities are all brand-new code and are unencumbered by AT&T license fees and other restrictions.

Minix Features

- System-call-compatible with V7 UNIX
- Kernighan and Ritchie compatible C compiler included
- Shell functionally identical to the Bourne shell included
- Full multiprogramming (fork+exec; background jobs in shell)
- Full screen editor inspired by emacs included
- Ability to read and write TOS disks
- Over 90 familiar utilities provided
- Over 100 library functions provided

- Floppy-only systems and hard-disk systems supported

- Full operating system source code (in C) included

- Source code for all the utilities (except C compiler) included

Partial List of the Minix Commands

ar as baddblocks basename cal cat cc cem
cg chmem chmod chown clr cmp comm
compress cp cpdir cpp cv date dd df diff
diskcheck du echo expr factor false find
fix fsck getlf grep gres head kill ld ln
login lpr ls make megarc mined mkdir
mkfs mknod more mount mv od opt
passwd pr printenv pwd readall readfs rev
rm rmdir roff sh shar size sleep sort split
stty su sum sync tail tar tee test time tos
touch tr treecmp true umount uniq
update uuencode uuencode wc

Partial List of the Minix Library Functions

abort abs access alarm atoi atol bcopy brk
call chdir chmod chown chroot cleanup
close creat crypt ctime ctype dprintf dup
dup2 exec exit fclose fflush fgets fopen
fork fprintf fputs fread freopen fseek fstat
ftell fwritegetc getegid getenv getuid
getgid getgrent getpass getpid getpwent
gets getuid gtty index ioctl isatty itoa kill
link lseek malloc mknod mktemp mount
open pause perror pipe popen putc puts
qsort rand read regexp regsub rindex
scanf setbuf setgid setuid signal sleep
sprintf stat stime strcat strcmp strcpy

srllen smcat stmcnp stmcpy stty sync
system termcap time times umask
umount ungetc unlink utime wait write

Newsgroup

There is a USENET newsgroup, comp.os.minix, concerned with MINIX. This group is gatewayed to the ARPANET, BITNET, etc. If you cannot read USENET newsgroups directly, you can get on the mailing list by sending a request to

info-minix-request@udel.edu

The group is very active, and well worth reading if you are interested in MINIX.

It is used for reporting bugs, fixing bugs, posting new software, asking and answering questions, and so on. At some point it may be necessary to split the group (IBM vs. Atari; source code vs discussion; ...) but for the time being, there are no plans to split it. Time will tell. There are archives of the messages that have been posted to comp.os.minix. For an information sheet telling about MINIX and the archives, send email to ast@cs.vu.nl or watch the newsgroup.

Documentation

There is a book describing MINIX in great detail, both how to use it and how it works inside. The book contains a highly annotated copy of the O/S code as an appendix (250 pages). This version is slightly out-of-date, but it is still quite usable. The bibliographic data on the book are as follows:

Title: Operating Systems: Design and Implementation

Author: Andrew S. Tanenbaum

Publisher: Prentice-Hall, Englewood Cliffs, NJ 07632

Date: 1987

ISBN: 0-13-637406-9 (hardback), 0-13-637331-3 (paperback)

Price: about \$40

The book is currently in the process of being translated into German.

There is also a paperback MINIX Reference Manual that is a subset of the book. It contains only the MINIX-specific information, not the general background on operating systems that the book contains. The reference manual is about \$35. There is also a package containing the disks and the reference manual combined going for about \$110.

The Atari disks come with a little booklet telling how to boot the system and how it differs from MINIX-PC (IBM version). Effectively it is a diff listing between MINIX-ST and MINIX-PC. It makes no attempt to repeat the 500 or so pages on MINIX from the book or manual.

Availability

MINIX is something of an intermediate form between AT&T UNIX and GNU. Unlike GNU, MINIX is not public domain. It is copyrighted by Prentice-Hall and is being sold by them. The price for the Atari disks in the U.S. is \$80 + shipping (somewhat higher abroad) and includes all the source code.

Unlike AT&T UNIX, the source code is readily available, and may be copied for bona-fide educational and research use. For example, a professor teaching a course on operating systems could legally buy the disks and then make copies for all his students. A limited amount of copying (say, no more than 3 copies per original) for friends is allowed.

If this gets out of hand, and Prentice-Hall decides that not enough have been sold, they will just drop the Atari and have future versions be for the IBM only. It is the intention that future versions be compatible with POSIX.

In the U.S., the software and books may be ordered from most bookstores or directly from Prentice-Hall in NJ. The ISBN number for the Atari software is 0-13-584392-8. Prentice-Hall's phone number is (201) 767-5937.

In the U.K., there are two dealers. The price in the U.K. is 88.50 pounds sterling + VAT

Prentice-Hall International
66 Wood Lane End
Hemel Hempstead England
Telephone: +44 442 231555

SDL Ltd
Unit 10
Ruxley Corner Industrial Estate
Sidcup Bypass
Kent England DA14 5SS

The distributors for Europe are listed below. The European price is \$110.40.

In Germany:

Steve Steinkrauf
Feldtorweg 24
D3406 Bovenden 1
FRG

In Holland:

Jos de Jong
Postbus 184
2100AD Heemstede
Holland

In Scandinavia:

Frank O'Donnell
P.O. Box 88
1371 Asker
Norway

In Spain and Portugal:

Deborah Worth
Appartado Numero 50672
Madrid
Spain

In Italy:

Jim Blaho
Via Manzani 50
50018 Scandicci
Florence
Italy

In Greece:

Vassilis Zahos
Kriconas 57
GR11634 Athens
Greece

In Turkey:

Atilla Gullu
Millinudafaa Cad 14/7
Kizilay Ankara
Turkey

MEMORY ALLOCATION

Mike Ditto of Ford Motor Company provides the following information to assist in the use of memory allocation using the malloc and free functions.

One is the most common symptoms of a corrupt memory free list is some type of memory access error occurring at some later time.

This may be caused by one of several problems:

Freeing memory (with free()) which is not currently allocated. This includes passing a null pointer to free() or free()ing the same block more than once.

Over-running the boundaries of an allocated block of memory. This is usually in the form of writing off the end of your block because you didn't request as many bytes as you really need.

Re-using memory which has already been freed or freeing memory which is still in use.

A bug in the malloc package. Pursue this option last, as it is the least likely.

Note that the types of errors that you, the programmer, can make, will not result in any sort of error until the next time you call malloc(). Some versions of free() can also die if the list is corrupt, but usually it's the next malloc() that dies.

Following is a technique to help track down malloc problems:

```
extern char *malloc();

#define MYMAGIC 0x2a /* Unlikely bit pattern */

char *mymalloc(nbytes)
unsigned nbytes;
{
    char *ptr = malloc(nbytes+9);
    if (!ptr)
    {
        fprintf(stderr, "mymalloc returning 0\n");
        return (ptr);
    }
    *((unsigned *)ptr) = nbytes;
    ptr += 8;
    ptr[-1] = MYMAGIC;
    ptr[nbytes] = MYMAGIC;
    fprintf(stderr, "mymalloc returning 0x%x\n", (long)ptr);
    return (ptr);
}

void myfree(ptr)
char *ptr;
{
    unsigned nbytes;
    fprintf(stderr, "myfree freeing 0x%x\n", (long)ptr);
    nbytes = *((unsigned *) (ptr - 8));
    if (ptr[-1] != MYMAGIC || ptr[nbytes] != MYMAGIC)
        fprintf(stderr, " (Freeing corrupt block!)\n");

    free(ptr - 8);
}

#define malloc    mymalloc
#define free      myfree
```

Put the above code in a header file and #include it if you want to enable the "verbose malloc". Redirect stderr to a file so you can study the messages. Here is an example output:

```
mymalloc returning 0x180c
mymalloc returning 0x182c
```

```
myfree freeing 0x180c
myfree freeing 0x182c (Freeing corrupt block!)
myfree freeing 0x180c
mymalloc returning 0x400c
```

As you can see in this example, there were two errors found. The second block malloced was corrupt when it was freed, and the first block was freed twice. Also, a potential error is that the third block was never freed, although this might be ok if the program intentionally exited with that memory still allocated.

Mike has used this technique many times to track down some pretty obscure bugs; he says that it's not fun but it's often the only way.

Naim Abdullah of the Department of EECS at Northwestern University provides the following suggestion on the same topic.

Ask malloc for the size being asked plus 4 extra bytes. Store the size in an int at the beginning of the block, advance the pointer by four bytes and return it to the caller.

In free(), back up the pointer by four bytes and get the number stored there as the size of the block. Give this pointer to the real free() to free the block.

Naim feels this approach is very elegant. Using it, he was able to find a memory leak (malloc without corresponding free) that he was looking for.

Following is some code from Karl Heuer (USENET karl@haddock.isc.com). It implements xmalloc(), xcalloc(), xfree(), xrealloc() and bytes_allocated(). Karl gave permission to use this code, so you may want to put it in your toolbox.

```
#define SLOP ((sizeof(unsigned int) + 7) >> 4)

extern void *malloc();
extern void *realloc();
extern void *calloc();
extern void free();
```



```

static unsigned int alloc_size =
0;

void *xmalloc(size)
unsigned int size;
{
void *p = malloc(SLOP + size);

if (p)
{
*(unsigned int *)p = size;
alloc_size += size;
p = (void *)((char *)p + SLOP);
}
return (p);
}

void *xcalloc(nmemb, size)
unsigned int nmemb;
unsigned int size;
{
void *p = calloc(SLOP + (size *
nmemb), 1);

if (p)
{
*(unsigned int *)p = size;
alloc_size += size;
p = (void *)((char *)p + SLOP);
}
return (p);
}

void xfree(p)
void *p;
{
if (p)
{
p = (void *)((char *)p - SLOP);
alloc_size -= *(unsigned int *)p;
free(p);
}
}

void *xrealloc(p, newsize)
void *p;
unsigned int newsize;
{
if (p)
{
p = (void *)((char *)p - SLOP);
alloc_size -= *(unsigned int *)p;
p = realloc(p, SLOP + newsize);
if (p)
{
*(unsigned int *)p = newsize;
alloc_size += newsize;
p = (void *)((char *)p + SLOP);
}
return (p);
}

unsigned int bytes_allocated()
{
return (alloc_size);
}

/* Now to test it */
#include <stdio.h>
main()
{
void *p1;
void *p2;

p1 = xmalloc(10);
p2 = xmalloc(99);
xfree(p1);
printf("%ld bytes still
allocated\n",

(long)bytes_allocated());

return (0);
}

```

PROGRAM TRACING UNDER UNIX V VERSION 4

Program tracing has been discussed at some length in previous chapters. It is a powerful technique used to assist in debugging and optimizing programs under systems supporting it.

Following is a description of a new method of program tracing, as docu-

mented by Roger A. Faulkner (USENET allegra!raf).

The `truss(1)` system call is based on AT&T's `proc(4)` process filesystem, invented by Tom Killian of Bell Labs research and extended and implemented for System V by Ron Gomes, with significant input from Roger Faulkner.

`truss(1)` can follow children created by `fork(2)`. It can trace a shell script of arbitrary complexity. This includes `spell(1)`, which runs an 8-member pipeline.

`truss(1)` can trace an arbitrary number of existing processes.

`truss(1)` allows the specification of which system calls to trace or exclude. `proc(4)` accepts a bit-mask to specify which syscalls to stop on. Untraced syscalls incur no overhead with `proc(4)`.

`truss(1)` does symbolic interpretation of syscall arguments, using `#define` names from relevant system header files. `truss(1)` has an option to turn off symbolic interpretation, for unredeemed hackers who must see the raw bits to be happy.

`truss(1)` (verbose option) shows the contents of structures passed by address to specified system calls. The contents are shown on output; values passed back from the operating system (like the `stat` structure from `stat(2)`) are displayed properly.

`truss(1)` shows all characters of any filename argument. This is related to the next item.

`truss(1)` always prints the first 16 bytes of a string argument in an unambiguous format. Also, `truss(1)` accepts an option to print the entire contents of the I/O buffer for `read(2)`s or `write(2)`s on specified file descriptors.

`truss(1)` optionally prints the argument and environment strings passed in each `exec(2)` system call.

`truss(1)` accepts an option to count system calls rather than showing them line-by-

line. `truss(1)` only counts those syscalls which are being traced; child process syscalls may be included in the counts.

`truss(1)` reports sleeping system calls as "sleeping ..." if they remain asleep for more than 2 seconds.

`truss(1)` reports the receipt of signals. By virtue of the `proc(4)` interface, `truss(1)` reports any machine fault which the process incurs when it is incurred, even if the associated signal is blocked.

`truss(1)` accepts options to trace or exclude specified signals or machine faults. `proc(4)` accepts a bit-mask of signals or faults to stop upon.

When `truss(1)` encounters an `exec(2)` of a set-uid or set-gid object (a process-tracing security violation), `proc(4)` forces it to give up and allow the process to continue unmolested. The `proc(4)` interface does a proper job of enforcing security without changing process behavior. If `truss(1)` is run as super-user, set-uid and set-gid processes can be traced with no problem.

The `proc(4)` mechanism is independent of the signal mechanism and does not interfere with its actions. A program using `proc(4)` can choose to trace signals or not; a signal is just one of the events a process can stop on; others are machine faults and syscalls. A process can be stopped without sending `SIGSTOP`. If the tracing process is killed, the traced process continues unaffected.

C PROBLEMS

Lloyd Zusman of Master Byte Software poses the problem below.

I'm looking for the fastest way to get the lowest power of two that is greater than or equal to a number.

Following is my first attempt:

```
unsigned int al(size)
unsigned int size;
{
```

```
    unsigned int try = 1;
    while (try && (size > try))
    {
        try <<= 1;
        return (try);
    }
```

This works and returns 0 if there is no power of 2 greater than the argument. But is there something that is faster? How about something involving a series of boolean operations?

Paul Raymond McMullin of Johns Hopkins University Applied Physics Laboratory makes the comments below.

Fast takes a strange meaning here... If you expect most of your inputs to this function to be "near" zero, the above function will probably be pretty good, taking 4 or five instructions (in the loop) per power of two contained in the answer. If your inputs to the function are distributed uniformly between 0 and 2^{31} (assuming a thirty-two bit word), you might try something like the following:

```
unsigned int al(size)
unsigned int size;
{
    unsigned int try = 1;

    if (size > 1<<16)
    {
        size = size >> 16;
        try = try << 16;
    }
    if (size > 1<<8)
    {
        size = size >> 8;
        try = try << 8;
    }
    if (size > 1<<4)
    {
        size = size >> 4;
        try = try << 4;
    }
```

```
    }
    if (size > 1<<2)
    {
        size = size >> 2;
        try = try << 2;
    }
    if (size > 1<<1)
    {
        size = size >> 1;
        try = try << 1;
    }
    if (size)
        try = try << 1;

    return try;
}
```

Notice that I've assumed that your compiler was smart enough to "constant fold" the shifted ones in the comparisons — for a stupid compiler you may have to do this by replacing them yourself. Also, notice that if your int sizes are longer than 32 bits you can either add additional cascades (e.g. "if (size > 1<<32)" etc, or change the first cascade into a while loop (which will make it portable to *any* length integers, masking off 16 bits at a time for *really big* numbers.

I've tested the program for the examples you gave above, but I didn't try MAXINT (or 2^{31} either).. I suspect that it'll return 0 for MAXINT like the other suggested algorithm, but haven't bothered to try it... I don't fool with unsigned numbers that much.

The program would be more clearly defined for the MAXINT case if you were asking for the power of two that contains the argument: you would initialize try to zero, and change all of the `try = try << x` statements to `try += x` statements.

It seems to me that what you should want for `al(1)` is 1, not two! (You seem to have forgotten that 2^{*0} is 1.) Fixing this case doesn't appear trivial at first, I'll leave it as an exercise for the reader...

I'm not sure why you defined al(0) to return 1 when al(2) returns 2; it seems to me that al(0) should be 0! If you change your mind, you'll have to handle the MAXINT case better, and add "else return 0;" as a completion for the "if (size) try = try<<1" statement.

Looking back over my code, it'll probably take 2 instructions per comparison that fails (e.g. small numbers falling through the first few cascades), and 4 instructions for each cascade that passes (two for the comparisons and two shifts), so for all but the cases where size < 32 my code will probably beat the first suggested implementation for speed.

Brian Beuning of AT&T makes the suggestions below.

Many bit problems can be solved by having an array of answers for a smaller number of bits and then breaking down the problem so it can be answered by an array lookup. With four range checks you can handle 32-bit input numbers, as follows:

```
short int bit8[256] = { 1, 1, 2,
4, ..., 256 };

unsigned int al(x)
unsigned int x;
{
    return ((x < 0x10000) ?
((x < 0x100) ? bit8[x] :
(bit8[x >> 8] << 8)) :
((x < 0x1000000) ?
(bit8[x >> 16] << 16) :
(bit8[x >> 24] << 24)));
}
```

With a different bit[] array, this same approach works for counting the number of bits set in a word.

Lloyd Zusman of Master Byte Software then answered his own question.

Some day I'll try to benchmark all the algorithms and see which one is *really*

the fastest on my system, but for now, I think I will use the following one that several people suggested:

```
unsigned int al(size)
unsigned int size;
{
    if (size)
    {
        size |= -size >> 1;
        size |= size >> 2;
        size |= size >> 4;
        size |= size >> 8;
        size |= size >> 16;
    }
    return (size + 1);
}
```

John Antypas (USENET jantypas@ucrmath.UUCP) poses the problem below.

In a graphics application we're writing at work, we have several functions for drawing lines, circles etc. We'd like to set up something like a vector table. If the vector table has a NULL in a spot, then the interpreter will use its internal routines instead of faster device-specific code.

As an example:

```
vectors[] =
{
    dev_line(),
    dev_box(),
    dev_circle(),
    dev_arc(),
    dev_text()
}
```

This says we have device specific code for all five functions. Yet when I try to copy function addresses, the compiler complains. How do I get the address of a function and put it into an array for calling later?

The reader may wish to consider the problem posed in the current section before reading the next section.

EXAMPLE C PROGRAM

Steven Weintraub at Motorola in Austin Texas ((512)440-3023) provides the following example program, which illustrates a use of function arrays.

```
#define COMMAND_LIST      1
#define COMMAND_ADD       2
#define COMMAND_QUIT      3
#define COMMAND_HELP      4
#define COMMAND_SHOW      5

char *buffer_place;
char buffer[150];

struct command_info
{
    char *name;
    short significant;
    int number;
    void (*function)();
};

void command_add();
void command_help();
void command_list();
void command_show();

struct command_info
command_table[] =
{
```

```

{"add", 1, COMMAND_ADD, command_add},
{"help", 1, COMMAND_HELP, command_help},
{"list", 1, COMMAND_LIST, command_list},
{"quit", 1, COMMAND_QUIT, NULL},
{"show", 1, COMMAND_SHOW, command_show},
{NULL, 0, 0, NULL}

};

void run_commands()
{
    char more;
    int linelength;
    struct command_info *this_command;

    more = YES;
    while (more)
    {
        printf("Command : ");
        linelength = readline(stdin, YES);
        if (linelength)
        {
            this_command = get_command(command_table);
            if (this_command->function)

                (*this_command->function)();

            else
            if (this_command->number == COMMAND_QUIT)

                more = NO;

            else
            {
                printf("Unknown command.\n");
                printf("Type Help for help\n");
            }
        }
        return;
    }

    struct command_info *get_command(command_table)
    struct command_info *command_table;
    {
        int count;
        char *this_c;
        char good;
        char more;
        char somemore;

```

```

    struct command_info *this_command;

    this_command = command_table;
    more = YES;
    while (this_command->name && more)
    {
        count = 0;
        this_c = this_command->name;
        buffer_place = buffer;
        good = YES;
        somemore = YES;
        while (good && somemore)
        {
            if (*this_c && *buffer_place &&
                *buffer_place != ' ')

                {
                    if (*this_c == *buffer_place)
                    {
                        count++;
                        this_c++;
                        buffer_place++;
                    }
                    else

                        good = NO;

                }
                else
                if (*buffer_place && *buffer_place != ' ')

                    good = NO;

                else

                    somemore = NO;

            }
            if (count >= this_command->significant && good)

                more = NO;

            else

                this_command++;

        }
        return(this_command);
    }
}

```

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

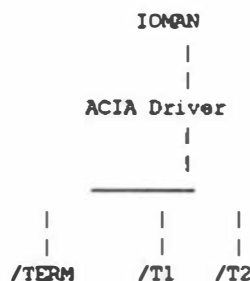
By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

INPUT/OUTPUT WITH WINDOWS

I thought this month we would get into talking about windows. The Color Computer using OS-9 Level II makes a fine example of windows. It is best to go back a ways and study Level I OS-9 on the Coco. It will reveal how the Coco screen has developed and why things are the way they are.

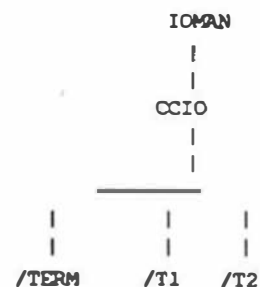
Level I OS-9, Version 1.00, had a 600 byte screen fixed in RAM. The screen was 32 X 16 in size. It displayed only uppercase characters using reverse video for the lower case. I really do not know anyone who stuck with it. A number of outside developers came up with alternates. Some were O-PAK, a graphics screen; WordPak, a plug in module that delivered a 80 X 24 screen; GO51, another graphics type screen from Stylo; and XSCREEN, yet another graphic type screen. They all did one thing — attempt to overcome the little green screen.

The internal structure was not much different from standard OS-9. Normally, the driver and descriptor structure for standard OS-9 are:



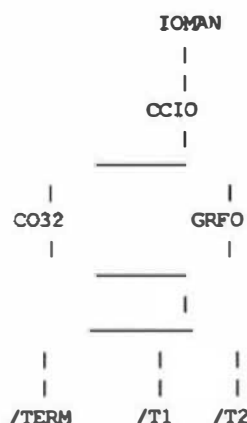
This diagram shows their relationship. Here the ACIA Driver is operating on /TERM, /T1 and /T2. There can be more, but this sufficient to display the workings. It is important to note that the driver is merely a I/O port. The terminal is external to the computer.

The Color Computer brought about an interesting change. The keyboard and screen were all part of the same computer. Actually the screen was your TV or monitor, but the VDG was from the computer and not really a concern of OS-9. For OS-9 Level I on the Color Computer we have:



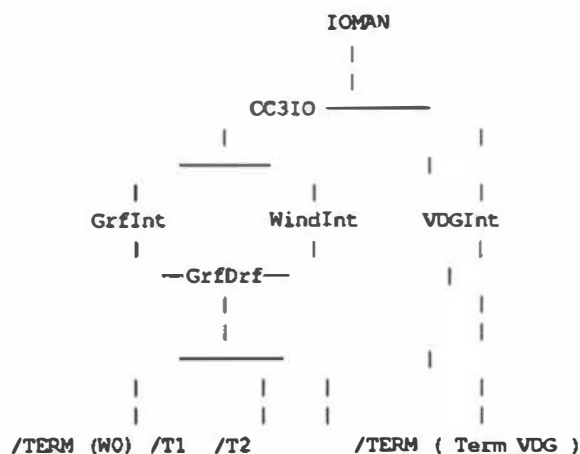
This is not a big different except that we have a clearly defined module that performs the keyboard and screen functions. The screen was located somewhere in memory mapped by the SAM (Synchronous Address Multiplexor). Later a new version 1 came up that allowed some extras such as a call to return the screen memory location was added, but the scheme was basically the same.

Eventually Version 2.00 came along it gave a new solution to the Color Computer input/output. It divided the function of CCIO into 3 parts. CCIO still existed to handle key board things, but output was allocated to modules, CO32 and CO80 which handle screen output. And graphics was handled through another module, GRFO. Separating graphics meant a memory savings when graphics was not being used.



This diagram gives a basic idea of how things are arranged. CO80 can be substituted for CO32 in this illustration. Important to note here is that things are becoming more flexible.

With OS-9 Level II, the design is very similar with some complications thrown in for good measure. In Level II a new screen system exists due to the addition of the GIME chip (and removable of the SAM). Now the old type screens are available as well as the new ones. The following diagrams illustrates the new arrangement.



Here we see a similar configuration, but a little more elaborate. GrfInt is the Graphics Interface. It handles window codes. WindInt is the window interface and it handles high-level windowing things, like pull down menus, borders, and scroll bars. It is available with the MultiView package from Radio Shack. GrfDrf is the text/graphics driver. VDGInt creates an environment like Level 1, Version 2.0.

These diagrams may not totally give the real picture. For example, keyboard I/O is done through CC3IO. Output however is handled by one of the Graphics interface. That is enough about the system. Now let's talk about windows.

The process to create a window is simple. Assuming that /W7 is in memory the following sequence will create a shell running in /W7

```

INIZ /W7
SHELL I=/W7 &
  
```

This sequence initialize /W7 and creates an shell running in it. The I option of the SHELL is something new and means it is an immortal process. It can be only terminated from the process. To move to the window, type the <CLEAR> key. A <SHIFT><CLEAR> will move back to the previous window. Once in the window anything can normally be done. One thing that I like is that when running multiple windows the process in each continues. Many systems do not work this way including some big ones.

Here is another interesting way to start a SHELL in /W7. From another window type:

```

SHELL <>>>/W7 &
  
```

I found this one can be terminated from the window that starts it.

If you have started an immortal process, it can only be terminated from it window. Enter:

```

EX
  
```

and the SHELL will terminate itself. Move to another window with the <CLEAR> key and enter:

DEINIZ W7

The window will go away.

A device window is one which can run some process. The previous examples were device window. Over a device window can exist overlay windows. An overlay window is used for displaying computer dialog. The big difference is a SHELL cannot be created in a overlay window, but it can be run in it. This month's program demonstrates this.

Listing 1 is a program called Pause. It prints a message to the screen and then waits for a key press to continue. This program is useful with procedure files. During the pause the user can do something like change a disk, or read a message. The following example changes the default foreground and background colors and prints the message in a screen centered window.

```
pause -f=2 -b=4 "Insert Disk B Into Drive /D0"
```

This causes the message to appear as:

```
Insert Disk B Into Drive /D0
Type Any Key To Continue!
```

Only the message will be centered in a window. If the -W option is used, the message can be listed to a standard terminal. It will appear pretty much as above.

The program Pause does have some things that can be improved. It does not know what type of screen it is in. GetStt can be used to get it and then the program can be a little smarter about what to size window to print and if it should use a window. Also, it echoes the key pressed to the screen which may not be desired. Using GetStt and SetStt to turn of and on keyboard echo would solve this. I leave these ideas to you. As written, Pause will work on a 24 x 80 screen.

I have some graphic calls in program to open and close a window, and turn of and on the cursor. The program can be linked to the CGFX.L available from Radio Shack. To compile the program use:

```
cc1 getopt.c -r
cc1 pause.c -r
cc1 pause.r getopt.r -l=/dd/lib/cgfx.l -f=pause
```

Let me tell you something interesting about CGFX.L. I received the library with my Development System, but there was no documentation. The documentation was in the Multi-View System, but it did not include the actual software. This strikes me as an unusual way to distribute software — splitting between two different packages.

Listing 2 is contains some C routines that basically do the same thing. They have similar calling format as the ones from CGFX.L. Include is owset(), owend(), curon() and curoff(). These will supply what is necessary to run PAUSE. To compile with the Listing 2 software use:

```
cc1 getopt.c -r
cc1 pause.c -r
cc1 cgfx.c -r
cc1 getopt.r pause.r cgfx.r -f=pause
```

I'll leave you with these listings. I have not commented on getopt.c which was from a previous issue of 68' Micro Journal. Getopt.c is available in a 3 volume set covering the listings of this column from the beginning to April 1988. The disks are \$9.95 each or all three for \$24.95. They can be ordered from South East Media at (615) 842-4600. I soon hope to have available the forth volume covering May 1988 to April 1989.

That's it for now. See ya soon.

LISTING 1

```

0000 /* *****
0001
0002 File: pause.c
0003 Date: 19 FEB 89
0004 Author: Ron Voigts
0005 Compile: OS9: make
0006
0007 *****
0008
0009 Function:
0010 1. pause() : prints a help message in
0011 a window and waits for key to be
0012 pressed.
0013 2. help() : provides help for pause().
0014 3. inkey() : returns key pressed.
0015 4. space() : prints spaces.
0016
0017 *****
0018
0019 Verison 1.0 RDV
0020 19 FEB 89
0021 Original
0022
0023 ***** */
0024
0025 #include "getopt.h"
0026
0027 #define TRUE 1
0028 #define FALSE 0
0029 #define STDOUT 1
0030
0031 int win_flag;
0032 int f_ground;
0033 int b_ground;
0034 char c[1];
0035 char *opt;
0036 char *list="?wf=b~";
0037
0038 main( argc, argv )
0039 int argc;
0040 char **argv;
0041 {
0042
0043 /* Set up default values */

```

```

0044 win_flag=TRUE;
0045 f_ground=0;
0046 b_ground=1;
0047
0048 /* Get options */
0049 optn=1;
0050
0051 while( (opt=getopt( argc, argv, list )) != 0 ) {
0052     if ( opterr == -1 )
0053         printf("Illegal option %c\n", *opt );
0054     else (
0055         if (*opt=='?')
0056             help();
0057         if ( toupper( *opt ) == 'W' )
0058             win_flag=FALSE;
0059         if ( toupper( *opt ) == 'F' )
0060             f_ground=atoi( optarg );
0061         if ( toupper( *opt ) == 'B' )
0062             b_ground=atoi( optarg );
0063     )
0064 }
0065
0066 if (win_flag) (
0067
0068 /* Create an overlay window */
0069     owset( STDOUT, 1, 5, 10, 70, 4, f_ground, b_ground );
0070
0071 /* Turn cursor off */
0072     curoff( STDOUT );
0073
0074 /* Print the option */
0075     printf("\n");
0076     space( (70-strlen(argv[optn]))/2 );
0077     printf("%s\n", argv[optn] );
0078     space( 25 );
0079     printf(" Type Any Key To Continue!\n");
0080
0081 /* Wait for keyboard response */
0082     inkey();
0083
0084 /* Turn cursor on */
0085     curon( STDOUT );
0086
0087 /* End the overlay window */
0088     owend( STDOUT );

```

```

0089
0090     ) else {
0091
0092     printf("%s\n", argv[optn] );
0093     printf("Type Any Key To Continue!\n");
0094
0095     inkey();
0096
0097     }
0098
0099 }
0100
0101
0102 /* Print n Spaces */
0103 space( n )
0104 int n;
0105 {
0106     while ( n-- )
0107         printf(" ");
0108 }
0109
0110 /* Return key pressed */
0111 int inkey()
0112 {
0113     char c[1];
0114
0115     while( getstat( 1, 0 ) == -1 )
0116         ;
0117     read( 0, c, 1 );
0118     return( c );
0119
0120 }
0121
0122 /* Print help information */
0123 help()
0124 {
0125     printf("Pause [-f=n] [-b=m] [-w] \"Message\"\n");
0126     printf("    f, changes foreground color to n (default 0)\n");
0127     printf("    b, changes background color to m (default 1)\n");
0128     printf("    w, turns off window\n");
0129     exit(0);
0130 }
0131

```

LISTING 2

```

0000 /* *****
0001
0002 File: cgfx.c
0003 By: Ron Voigts
0004 Date: 29 FEB 89
0005
0006 *****
0007
0008 Fucntion:
0009 1. owset() : open an overlay window
0010 2. owend() : close window
0011 3. curon() : cursor on
0012 4. coroff() : cursor off
0013
0014 *****
0015
0016 Version 1.0      RDV
0017 19 FEB 89
0018 Original
0019
0020 ***** */
0021
0022 /* Creates an overlay window of
0023 size szx x szy starting at position
0024 cpx, cpy. Foreground and background
0025 colors are fprn and bprn. If svx is 0,
0026 area under window is not save, if it is
0027 1 it is saved. */
0028
0029 owset(path,svs,cpx,cpy,szx,szy,fprn,bprn)
0030 int path,svs,cpx,cpy,szx,szy,fprn,bprn;
0031 {
0032
0033 /* Create buffer for 'write' string */
0034 char b[9];
0035
0036 /* Initialize the buffer */
0037 b[0]=0x1B;
0038 b[1]=0x22;
0039 b[2]=svs;
0040 b[3]=cpx;
0041 b[4]=cpy;
0042 b[5]=szx;

```

```

0043     b[6]=szy;
0044     b[7]=bprm;
0045     b[8]=fprm;
0046
0047 /* Write the buffer */
0048     if ( write( path, b , 9 ) == -1 )
0049         return( -1 );
0050
0051     return( 0 );
0052
0053 }
0054
0055
0056 /* Closes an existing window */
0057 owend( path )
0058 int path;
0059 {
0060
0061 /* Create buffer for 'write' string */
0062     char b[2];
0063
0064 /* Initialize buffer */
0065     b[0]=0X1b;
0066     b[1]=0X23;
0067
0068 /* Write the buffer */
0069     if ( write( path, b, 2 ) == -1 )
0070         return( -1 );
0071
0072     return( 0 );
0073 }
0074
0075
0076 /* Turns cursor on */
0077 curon( path )
0078 int path;
0079 {
0080
0081 /* Create buffer for 'write' string */
0082     char b[2];
0083
0084 /* Initialize buffer */
0085     b[0]=0X05;
0086     b[1]=0X21;
0087

```

```

0088 /* Write the buffer */
0089     if ( write( path, b, 2 ) == -1 )
0090         return( -1 );
0091
0092     return( 0 );
0093 }
0094
0095
0096 /* Turns cursor off */
0097 curoff( path )
0098 int path;
0099 {
0100
0101 /* Create 'write' buffer */
0102     char b[2];
0103
0104 /* Initialize buffer */
0105     b[0]=0X05;
0106     b[1]=0X20;
0107
0108 /* Write the buffer */
0109     if ( write( path, b, 2 ) == -1 )
0110         return( -1 );
0111
0112     return( 0 );
0113 }

```

LISTING 3

```

0000 /* *****
0001
0002     Name: GETOPT
0003     By: Ron Voigts
0004     Date: 25-MAY-87
0005
0006     *****
0007
0008     Function:
0009     This function examines the argument list
0010     returning a pointer to the option and
0011     its argument. A null string is pointed
0012     to if the option has now argument.
0013
0014     *****
0015
0016     Version 1.00

```



```

0017 Original.
0018
0019 ***** */
0020
0021 #define TRUE 1
0022 #define FALSE 0
0023
0024 char *optarg; /* Option argument */
0025 int optn; /* Next option */
0026 int opterr; /* Error status */
0027
0028 char *getopt( c, v, optlist )
0029 int c; /* argument count */
0030 char **v; /* argument vector */
0031 char *optlist; /* option list */
0032
0033 {
0034     int isoption; /* option flag */
0035     int hasarg; /* option argument flag */
0036     register int i; /* useful index */
0037     char *opt; /* option pointer */
0038     char *t; /* argument pointer */
0039     static char *null = '\0'; /* null string */
0040
0041 /* Set up the null string for 'optarg' */
0042     optarg = null;
0043
0044 /* Set up the error return status */
0045     opterr = 0; /* No errors */
0046
0047 /* Set up the argument */
0048     t = v[optn];
0049
0050 /* We are at the end of the argument list */
0051     if ( (optn == c) || (t != '-' ) || ( *t == '-' && *(t+1) == '\0' ) )
0052         return( 0 );
0053
0054 /* We can set the option */
0055     opt = t+1;
0056
0057 /* Check if we have an option with an argument */
0058     isoption = FALSE;

```

```

0059     hasarg = FALSE;
0060     for ( i=0; i<strlen(optlist); i++ )
0061         if ( toupper(*(t+1)) == toupper(optlist[i]) ) {
0062             isoption = TRUE;
0063             if ( optlist[i+1] == '=' )
0064                 hasarg = TRUE;
0065         }
0066
0067 /* If this is not an option then return with error */
0068     if ( !isoption )
0069         opterr = -1; /* illegal option */
0070
0071 /* Now we check and set up the argument */
0072     if ( hasarg ) {
0073         if ( *(t+2) == '\0' )
0074             if ( optn < c-1 )
0075                 optarg = v[++optn];
0076             else
0077                 opterr = -2; /* Missing option argument */
0078         else
0079             optarg = t+2;
0080         if ( *optarg == '-' )
0081             optarg++;
0082     } else
0083         if ( *(t+2) != '\0' )
0084             opterr = -3; /* Argument not expected */
0085
0086 /* Now we have an argument and option */
0087     optn++; /* Adjust the next pointer */
0088     return( opt ); /* Return the option pointer */
0089
0090 }
0091

```

LISTING 4

```

0000 extern char *optarg; /* Option argument */
0001 extern int optn; /* Next option */
0002 extern opterr; /* Error Status */
0003

```

FOR THOSE WHO NEED TO KNOW

68 MICRO
JOURNAL™

COME ON IN THE WATER'S FINE

PLUS FIVE

1968 Innerbelt Business Center Drive
St. Louis, Missouri 63114
314/426-3900

1.0 INTRODUCTION TO PLUS FIVE

Plus-Five specializes in MUMPS and has done so for seven years. MUMPS was originally implemented as an operating system, database, and programming language environment. In 1982, Plus Five separated the operating system portion from the original implementation. The result was a very powerful programming language and database tool, which is capable of being used on another operating system (originally UNIX). In the past seven years, Plus Five has ported MUMPS to over 50 different manufacturers' platforms, from an Amdahl 5870 to PC ATs. Plus Five's MUMPS is a full implementation of the 1984 ANSI Standard, meets or exceeds the Federal Information Processing Standard (FIPS), and has several additional powerful extensions. MUMPS has become so well established that OEMs are frequently required to have MUMPS available on their systems in order to respond to governmental requests for proposals (RFPs). At least one OEM was unable to deliver 100 million dollars worth of hardware until MUMPS was made available on it.

When Plus Five was first approached about porting MUMPS to OS-9, we had plenty of reasons to hesitate. After all, our forte was UNIX, not OS-9, and we had little experience with real-time operating systems. And, MUMPS is not your average weekend project - we're talking heavy duty here, over 64,000 lines of code! However, the concept was too exciting and the challenge too tempting - we could not resist. Imagine all the existing MUMPS applications¹ and the power and programming efficiency of MUMPS combined with the graphics and real-time capabilities of OS-9.

We obtained some inexpensive OS-9 hardware and just sort of dove in. Well, we want to say thanks to Microware² and to Ultrascience³. Microware's excellent C, powerful OS-9 C source debugger, and experienced support team made the programming task a pleasure. Ultrascience's rugged and nicely implemented OS-9 coprocessor for IBM personal computers and clones, the PC68K1/MEMIOX, provided us with affordable machines for development and promotion. The PC68K1 permits us to run OS-9 and DOS concurrently and to move data between the two operating systems.

¹ MUMPS SOFTWARE SOURCES₁, published by the MUMPS Users' Group, 4321 Hartwick Rd., Suite 510, College Park, MD 20740, Phone: 301/779-6555

² Microware System Corporation, 1900 N.W. 144th Street, Des Moines, IA 50322, Phone: 515/224-1929

³ Gibbs Laboratories, Inc., Ultrascience Division, 1824 Wilmette Ave., Wilmette, IL 60091, Phone: 312/256-0080

FYI:

68 Micro Journal excerpted the following introduction from *EXPLORE THE WORLD OF MUMPS*, published by the MUMPS Users' Group, 4321 Hartwick Rd., Suite 510, College Park, Maryland 20740.

WHAT IS MUMPS?

MUMPS is a multi-purpose, interpreted or compiled, high-level programming language designed for interactive data management applications. Its string-handling features and hierarchically structured database suit it for medical, business and word-processing applications where large, complex files must be designed, constructed and maintained.

MUMPS was developed at Massachusetts General Hospital in Boston in 1966. The MUMPS acronym stands for Massachusetts General Hospital Utility Multi-Programming System. The language is used in medical, commercial, industrial, and academic applications worldwide, in addition to a growing number of personal computer configurations.

MUMPS is an American National Standard Language, which means that all implementations of Standard MUMPS use the same commands, functions, and operators. This enables programs to be transported between different computers running standard MUMPS. Users are not restricted to a single hardware or software vendor. If a system grows, it can be easily moved to a larger computer. This also means users can share applications software.

MUMPS is a FEDERAL INFORMATION PROCESSING STANDARD (FIPS). The Institute for Computer Science and Technology develops Federal Information Processing Standards (FIPS) for programming languages when there are significant benefits for Federal users, and when technically sound specifications exist. Federal departments and agencies may select MUMPS as their language of choice.

WHO IS USING MUMPS?

MUMPS is being used for virtually every type of health service, scientific, technological and corporate data processing system. It is most often used for on-line, interactive applications where immediate data retrieval and prompt response are important. Medical and business organizations across the globe use MUMPS—organizations such as the Smithsonian Institution, the Department of Defense for its Composite Health Care System, the Veterans Administration Hospitals, the U.S. Navy Occupational Health Program, trust banking institutions, a major American airline company, credit unions, universities, and the Swiss government.....

WHY SHOULD YOU CONSIDER MUMPS?

MUMPS is simple to learn and powerful to use. It gives the novice programmer the tools to develop professional programming applications. Experienced programmers find that the language facilitates the solving of complex data management problems. Its simple structure and interactive nature help the programmer to create and de-bug programs quickly.....

Studies show that MUMPS significantly enhances programmer productivity over other languages. MUMPS programmers spend only 8-12% as much time for system development. In addition, a typical MUMPS program requires only 10% as many lines of code as other languages.

Another important attribute is program portability. Users can move MUMPS programs between different computer hardware and MUMPS implementations.

2.0 ULTRASCIENCE PC68K1 AND MEMIOX

The OS-9 PC68K1 coprocessor board is a full length board that plugs into a standard PC/XT/AT style bus. It is available in 68000 or 68010 versions which run at 10 or 12.5 MHz. 1 Mb of zero wait-state DRAM, 2 vectored serial ports (DB9), a parallel port, battery backed up time-date chip, Professional OS-9 and extensive diagnostic utilities and device drivers are standard. 8 K of battery backed up SRAM is optional. The MEMIOX expansion board couples directly to the PC68K1, as a daughter board, by means of a flexible cable. It provides another 1 Mb of zero wait-state DRAM and up to 10 more vectored serial ports (RJ45).

Plus Five requested the convenient Ultrascience "Click" cables because we wanted to get started with the absolute minimum of effort. The cables use 8 conductor, telephone-style, RJ45 terminated cable, which is fully shielded. DB9 and DB25 adapter-connectors are available in both sexes to provide null modem, terminal, printer, and COM port connections. "Click" connectors made our cabling a SNAP and we highly recommend them for any type of computer installation. Within minutes our first PC68K1 was installed, connected to a number of terminals, and ready for power-up.

With a flick of the power switch, the PC proceeded to boot and the PC68K1 came alive, issuing an impressive self-test message to the terminal attached to port "0". We followed the simple and straight-forward installation instructions provided with the PC68K1 DOS diskette which loaded the necessary utilities onto the DOS partition. We then loaded the OS-9 software, as instructed, and in a few seconds were welcomed to the world of OS-9.

The PC68K1 DOS/OS-9 software makes use of a 1K, memory mapped, dual ported DRAM which buffer-couples the 68K bus of the PC68K1 and the PCibus. By means of this high-speed interface, OS-9 can share the hardware and software resources of the PCbus. The monitor (monochrome, CGA, EGA, or EVGA) can be hot-keyed back and forth between OS-9 and DOS (screens are not lost), and OS-9 utilizes the PCbus diskette (DSDD or DSHD 5-1/4" and 3-1/2"), hard disk (ST506 or SCSI), and tape (SCSI) drives. OS-9 can also access the PCbus COM port and printer ports as standard OS-9 devices. The drivers and programs which make all this possible are provided standard with the PC68K1.

Deciding how to partition hard disk capacity between OS-9 and DOS took some deliberation; not because it was especially difficult, but because the options were plentiful. OS-9 partitions can be installed on up to 4 drives, including the DOS boot drive, and any of these drives can be the OS-9 boot partition. Ultrascience has automated the installation of OS-9 with a command file which takes over once the user has created the required "OTHER/122" type partition with DOS hard disk partitioning software. It appears that a variety of partitioning software will perform this function, but we used the recommended Disk Manager product from Ontrack4 (provided by Ultrascience) and encountered no difficulty.

The OS-9 driver and related software for the PCbus monitor permits the monitor to function as a Link Model 125 non-embedded terminal with special color features. Though we encountered some bugs in the first release of the driver, the current version seems to be bug free. The OS-9 driver and related software for the PCbus keyboard currently supports about 90 combinations of ALT, CTRL, SHIFT, and NO SHIFT with function keys and normal keys to serve as special purpose function keys. The key combinations can be programmed, and lead-in and trailer characters can be changed or suppressed. Altering the output of function keys to suit a series of screens or applications has been automated.

Plus Five has made good use of PC68K1 OS-9 programs for moving data back and forth between OS-9 and DOS disks. At this time, Ultrascience does not support wildcarding for this purpose. It would be a nice feature, but it is also an understandably complex task, which might cause serious problems if it were not implemented flawlessly. Software for spawning DOS programs from OS-9, and for using the 1K dual ported DRAM for passing data between DOS and OS-9 programs, is also provided. We have yet to explore the potential of this latter resource, which is certainly of importance to process control developers.

The standard PC68K1 software also comes with an impressive array of utilities, designed for diagnosing and servicing boards in the field. In most cases, diagnosis is chip specific. We have not had any need to repair our boards, but it appears that the products would be very easy to support, especially since all the active components are socketed in quality gold sockets.

Ontrack Computer Systems, Inc., 6200 Bury Drive, Eden Prairie, MN 55346

Link Technologies, Inc. (A Wyse company), 47339 Warm Springs Blvd.,
Fremont, CA 94539

3.0 MICROWARE OS-9

At some point in the future, Plus Five hopes to put together an article, detailing differences between UNIX and OS-9; that is, if someone has not already written one! The differences are not trivial; however, in our case, they were certainly manageable. But, creating software is usually easier than debugging it; and in this regard, we would be at it still were Microware's fantastic C debugger not available. The debugger allowed us to eliminate those infamous "GOTCH-YAs" in record time, and well within budget, by pin-pointing the exact code that created an error. This is a tool no developer should be without.

Plus-Five is also impressed with the hardware portability of OS-9. We cloned our OS-9 MUMPS from the PC68K1 PCbus platform to a 25MHz, Motorola 147 (68030) CPU based, VMEbus system in a few hours. That was a real treat, and we see no reason why we should experience any special problems installing OS-9 MUMPS on any properly implemented port of OS-9.

Another OS-9 feature, which Plus Five found to be superior as compared with UNIX, is its interprocess control (IPC) feature. Efficient interprocess communication permits the development of significantly more sophisticated applications with much less effort. Consider what happens when a conventional operating system and software are monitoring and collecting data from a dozen or more scientific instruments and an emergency situation arises on one of these instruments (i.e., the temperature threshold is exceeded). Jumping to an unscheduled emergency routine could easily result in loss of data from one of the other inputs. Avoiding such a pitfall can be complex when one is writing software for a conventional operating system. Using OS-9's IPC feature, each of the scientific instruments could have its own monitoring and data collection process, which would operate independently of all others, and each of these processes could communicate with one another and with independent emergency routines. Data collection and equipment monitoring activities would not be interrupted by unscheduled emergencies. The importance of IPC to medicine, where MUMPS is already very strong, and to industry, where MUMPS is gaining acceptance rapidly, is obvious.

Leads would be greatly appreciated.

4.0 CONCLUSION

MUMPS benefits substantially from the powerful features of OS-9. Plus-Five is impressed with OS-9, with Microware documentation and support, and with the Ultrascience products. We applaud the Ultrascience OS-9 PC68K1 coprocessor as the ideal means to spearhead an increased penetration of OS-9 into the classroom, development lab, and thence into the broad marketplace. The cost-effectiveness of the PC68K1 compared to LANs, its compatibility with PCbus machines, and the portability of OS-9 applications to larger machines gives applications developers a marvelous opportunity.

KEYWORDS:

MEMIOX (Ultrascience)
MUMPS (OS-9 and UNIX)
OS-9 (Microware)
OS-9 MUMPS (Plus Five)
PC68K1 (ULtrascience)
UNIX MUMPS (Plus Five)

+++

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Some Thoughts on a 68xxx ROM Monitor

Karl Lunt
7349 W. Canterbury
Peoria, AZ 85345 (602)-878-0305

My newest computer is a 68010 board I purchased surplus for \$40. One of the first things a user of such a surplus board must do is to rewrite the original ROMs so the board will do something useful for its new owner. That means designing and writing a new ROM monitor.

I had already written several ROM monitors for my 6809 systems and learned several things that I wanted to change in the 68xxx monitor I was preparing to write.

The new ROM had to support a string output routine much more sophisticated than the PSTRING utility normally found in the 6809 monitors. Specifically, I wanted the string utility to support multiple arguments and formatted hex/decimal output.

The monitor had to permit on-the-fly changes to the primitive input and output routines. This was necessary to permit either of the two serial ports to use the monitor as needed. Ideally, changing from serial port 0 to serial port 1 should be no more difficult than changing a couple of RAM vectors.

The new monitor had to use NO local RAM variables. This was to permit more than one user to be running the monitor at the same time, in turn allowing the monitor to work in a multi-user fashion. As it turned out, the only local RAM variables used by the monitor are the I/O vectors in RAM, described above, and a memory-test flag used at RESET time.

The monitor had to provide some simple system checks upon power-up or reset, but also had to be smart enough to recognize when a user program was already installed in memory and not overwrite that program with a memory check just because someone hit the RESET switch.

This monitor also had to be callable from a user program. This means that a user should be able to execute a JSR MONITOR to enter the monitor from his program, make full use of the monitor's capabilities, then EXIT back to his program and continue as if never having left the program.

A monitor that does all of the above, plus considerably more, is presently running on my 68010 board. I will make the source listings available to 68MJ readers for use in their own systems or for any other non-commercial use they might have. Credit to different sources of information appear throughout the source listings; if you use this code, please leave the notices intact so the original thinkers get the credit they deserve.

The 680x0 code for the monitor was developed on a PC/XT compatible running the 2500 A.D. 68000 cross assembler and linker system. I used Sidekick as my text editor, principally because it is fast, RAM-based and I already knew how to use it. In the early stages of development, I had to burn different versions of the monitor into ROM, install the ROMs in the board, try the software, then repeat the cycle as necessary. The current version of the ROM supports downloading of S19 records, considerably easing the development of new software or monitor extensions. I have been using Crosstalk to communicate between the host PC and the target 68010 system, but just about any good PC public-domain modem program should work fine.

The first order of business is the formatted string output routine, PRINTF. The code for this routine can be found in the book "68000 Assembly Language," by Stanley and Kranz (published by Addison-Wesley). This book is an absolute gold-mine of information for the 680x0 programmer; I consider it a "must-have" book and have nearly destroyed my copy from overuse. (Don, how about a review of this book?) (o.k. Karl, we'll be looking forward to it - DMW)

The PRINTF function, like its C namesake, permits the output of formatted string and numeric information. All arguments to PRINTF are placed on the stack and must be popped from the stack by the calling program after the return from PRINTF.

Each call to PRINTF must have at least one argument; the LONG address of the formatted string to be printed. If this formatted string requires any arguments of its own (such as a decimal number to be printed in the body of the string), those arguments must also be pushed onto the stack. These secondary arguments must be pushed in reverse order of need, prior to pushing the address of the formatted string. A short example might make this more clear.

Consider a formatted string to print the amount of RAM available to a user and the first address of that RAM. It would make most sense if the first value was displayed in decimal and the second value in hex. An example of how you might want the display to appear could be:

AVAILABLE RAM: 129300 BYTES FIRST RAM LOCATION: \$42000

The following segment of 680x0 code, using the PRINTF function, will do just that.

```

MOVE.L    A0, -(A7)      FIRST RAM LOC IN A0 TO
                           STACK
MOVE.L    D0, -(A7)      AMOUNT OF RAM IN D0 TO
                           STACK
MOVE.L    #FMTSTRG, -(A7)  PUSH ADDRESS OF STRING
JSR        PRINTF         DISPLAY THE STRING
ADD.L     #12, A7         PULL 12 BYTES FROM STACK
BRA       —              REST OF PROGRAM FOLLOWS

FMTSTRG:
BYTE      $0D, $0A
BYTE      'AVAILABLE RAM: %D
BYTES     '
BYTE      'FIRST RAM LOCATION: %X'
BYTE      $0D, $0A
BYTE      0              REQUIRED NULL TERMINATOR

```

As you can see, using the PRINTF routine is only slightly more complicated than the 6809 PS*TRING routine, and far more powerful. Besides handling LONG decimal and LONG hex values, the PRINTF routine can handle a variety of other formatted data. The full list is:

```

WORD decimal
%d displays 16-bit signed decimal LONG decimal
%D displays 32-bit signed decimal WORD decimal
%u displays 16-bit unsigned decimal LONG decimal
%U displays 32-bit unsigned decimal WORD decimal
%x displays 16-bit hex LONG hex
%X displays 32-bit hex string
%s displays null-terminated string character
%c displays single literal char

```

Although the routine is named PRINTF, its formatting convention differs significantly from the C version of PRINTF; full details on how the routine works can be found in "68000 Assembly Language" and by looking through the examples in the ROM monitor code.

The implementation of I/O vectors is based somewhat on the method used by FLEX for redirecting I/O. The way the monitor handles the I/O, however, is more flexible.

All calls to the general purpose I/O routines are to ROM entry points named GETC, PUTC and KEYPRSD. For example, if your program wants to output a character to the terminal, it pushes the character (as a WORD) onto the stack and JSRs to PUTC. When control returns from PUTC, your program must then pop the character off of the stack before continuing.

Each of these primitive routines, however, is routed through a RAM-based vector to the appropriate I/O routine, depending on which user port is active at the time. For example, if user port 0 is active, GETC will jump through a RAM vector that is aimed at GETC0, the routine that gets a character from serial port 0.

Although the code for both GETC and GETC0 reside in ROM, the link between the two is in RAM and may be modified at any time. This link could just as easily point to GETC1, the ROM-based routine that services serial port 1. In this case, GETC would return

a character input from port 1. A full set of GETC, PUTC and KEYPRSD routines exist in ROM for both of the serial ports.

This concept of RAM-based links for the primitive I/O routines can be extended even farther. If a user's program wants to talk to a printer (as an example) or a virtual terminal, the RAM links to the I/O functions can be changed to permit the user's interface routines to run. Because PRINTF and all other higher-level monitor routines use only the general GETC, PUTC and KEYPRSD entry points, the full monitor will run on any set of drivers installed with the RAM links.

But how do you write a monitor that uses NO global RAM variables? If you have never programmed on the 680x0 before, you are in for a real treat. Motorola provided the programmer with the LINK and UNLK instructions, specifically designed for creating stack-based variables. The structure created by these instructions is called a frame and it is crucial to much of the power in this ROM monitor.

The customary way for a routine to handle variables used to involve setting aside a small, fixed area of RAM for storage of things such as flags, counters, pointers and other data structures. If you tried to use the same piece of code to service two different users, however, one user's variables would get overwritten by the other's values, causing the routine to lose track of what was going on.

The answer to this problem is to put the variables in the stack. Since each user would (hopefully) enter the routine with a different stack pointer value, the variables associated with that user would be safely stored out of the way and available only to the proper user.

This, however, leads to a different problem. Upon entry to a routine, the item on the top of the stack is the return address (something your routine is liable to need again soon). Behind that address (going upwards in memory) may be arguments needed by the routine itself. Behind those arguments will be other return addresses and eventually the "bottom" of the stack. This means that a routine cannot arbitrarily store local variables and data on the stack using a positive displacement (that is, using addresses higher than the current stack pointer).

Well, how about going downwards (using negative displacement) on the stack? While it is possible to store and retrieve variables using the stack pointer plus some negative offset, this technique is doomed to failure as soon as interrupt or exception activity begins. The reason for this should be obvious; when an interrupt occurs, the first thing the CPU does is push vital information onto the stack and race off to service the interrupt. If this interrupt occurs in a routine that is storing data on the stack using negative displacements, that data will occasionally be trashed.

Enter, then, the LINK instruction. LINK "freezes" the stack by first copying the current stack pointer value into a selected register (by convention this is usually A6), then moving the stack pointer downwards in memory a specified amount. This has the effect of reserving a block of the stack for the routine's use RELATIVE TO

THE A6 REGISTER. Now, references to stack-based variables relative to A6 can be safely made, regardless of interrupt activity. When it comes time to exit the routine, the UNLK instruction repairs the stack (it even restores the selected register) and leaves the return address on the stack, ready for the subsequent RTS.

The best way to get a handle on the LINK instruction and the concept of stack framing is to study the examples in the monitor code and to read the text in "68000 Assembly Language." This is an extremely valuable programming tool and is worth the time it takes to understand it.

The ROM monitor makes heavy use of stack framing. In fact, the only global RAM variables used by the monitor are the I/O links discussed above and a couple of flag LONGs used to determine the course of action to take on reset. The entire monitor, then, can be thought of as a huge subroutine. Any user can execute the monitor by simply treating it as a subroutine and JSR'ing to it. All of the monitor's text buffers, flags and other data will be safely tucked away in the calling program's stack.

The ROM monitor performs a few simple chores upon reset. It tests the mapping RAM (used to assign physical RAM pages to logical addresses) and it tests all of available RAM before it configures the RAM-based serial links and announces its presence.

But this testing of RAM, even though it is important the first time you apply power, can be a real pain if the program you are testing runs away, forcing a RESET. All of your patching and testing can get wiped out by the RAM check, costing you a lot of work.

To prevent this from happening, the monitor writes a key value in a specific location of RAM after it has successfully tested memory. On the next reset, it first checks this location to see if the key value is still there. If it is, the RAM test is skipped, leaving the user's program intact.

As I have already mentioned, the monitor's heavy use of stack framing permits it to be treated as a giant subroutine. In fact, the code itself reveals that there is a huge routine called MONITOR\$ and a very small outer loop that exists solely to call MONITOR\$. If you want to access the monitor from within a user program, you need only JSR to the ROM address named MONITOR. At that time the active port will display the monitor's prompt and you may enter any legal monitor commands. If you then use the monitor's EXIT command, the monitor will execute a RTS, returning you to your original program.

This provides some very powerful capabilities to someone trying to develop software on such a system. Adding a special keyboard sequence into your program code (for example) permits you to activate the monitor, view or modify memory as desired, then return to your program to continue with your testing.

This is only a part of the resources available to a developer using this monitor ROM. Other goodies in the code include the obliga-

tory memory dump (DU), the ability to upload Motorola S19 records with an optional address offset (RL), an imbedded CASE support routine available to a calling program, and much more.

If there is enough interest in this ROM monitor, I can continue this discussion in subsequent issues of 68MJ. For example, I already have code running (not yet in the monitor ROM) that allows more than one user to execute the monitor simultaneously, providing true multi-user power at the monitor.

Editor's Note: If you want more of this please let me know, or communicate directly with the author. With the availability of "bargain" and surplus hardware, this type of information is like a lamp-post in the night! Takes me back a few years, how 'bout you?

DMW

• MONITOR

- * Main routine in the 68010 EPROM.
- * Note that there is no ORG statement in this block. It should be
- * LINKed to appear at address \$800100, as that is where the COLDS
- * vector is aimed.

```
EXTERNAL PRINTF, SINIT, PUTC, GETC, GETLINE
EXTERNAL DUMPMEM, FILL, SWEEP
EXTERNAL PROMPT
EXTERNAL CASE, UPPER
EXTERNAL ACTRAM, TESTRAM
EXTERNAL CHANGE
EXTERNAL S19, GO
EXTERNAL MONITOR, WARMS1
EXTERNAL DUMP_REG$
```

```
PUBLIC WARMS$, COLDS$, PROCESS$, MONITDR$
PUBLIC WARMS1$
```

```
BUFFER EQU -256      STACK-BASED TEXT BUFFER
*                               STACK MUST ALLOW FOR 256
CHARS
*                               REFER TO MONITOR
*                               (BELOW) FOR USE
```

```
MON_SP: EQU SDF0      STACK POINTER
*                               USED BY MONITOR.
*                               THIS IS THE
*                               VALUE WRITTEN TO A7
*                               BY BOTH COLDS AND
*                               WARMS1.
```

```
COLDS$:
    MOVE.W    #$3700, $450000    LEDS:      _ _ G _ R
    MOVE.L    #$400000, AD       POINT AT START
                                OF MAP RAM
                                SET A COUNTER

    MOVE.W    #$3FF, D0

MAPTEST:
    MOVE.W    #$01A5, {A0}       WRITE A WORD
    MOVE.W    {A0}+, D1          READ IT BACK
    AND.L     #$01FF, D1         MASK OUT
                                CONTROL BITS
```


| | | | | | |
|--|--------------------|-----------------------|------------|----------------------------------|-----------------------|
| CMP.W | #\$01A5,01 | NOW TEST IT | EXITMSG: | BYTE | \$0D,\$0A,\$DA |
| BNE | RAMFAIL | BRANCH IF | BYTE | 'An EXIT back to the ROM Monitor | |
| | | FAIL | | is' | |
| DBF | D0,MAPTEST | COUNT THIS | BYTE | ' pretty pointless, you know.' | |
| | | WORD | BYTE | \$0D,\$0A | |
| BRA | STARTUP | BRANCH IF ALL | BYTE | 0 | |
| | | GOOD | | | |
| * * Main routine for the 68010 ROM monitor. * * Normal entry is from a COLD start following reset. * Alternate * entry is from a user program via the WARMS (or WARMS1 entry. * * A user program may also enter via the MONITOR jump vector in * the ROM jump table. Entering through this point permits a user * program to activate the monitor with a special se- quence from * within the program, use the monitor as if it had been activated * by a warm-start, then return to the user program with an EXIT * monitor command. Entry by this technique should be with a JSR. * Leaving the monitor this way will return to the calling program * with ALL registers preserved, though the CCR is not saved. * | | | | | |
| RAMFAIL: | | | | | |
| BRA | RAMFAIL | DIE HERE WITH | | | |
| | | LEDS ON | | | |
| STARTUP: | | | | | |
| MOVE.W | #\$3E00,\$450000 | LEDS: R _ G _ _ | | | |
| MOVE.W | #\$2000,\$400000 | SWITCH ON | | | |
| | | LOWEST 4K BLOCK | | | |
| MOVE.L | #MON_SP,A7 | PUT STACK IN | | | |
| | | LOWEST RAM | | | |
| MOVE.L | #\$A5A5A5A5,\$0FFC | TEST A LOCATION | | | |
| CMP.L | #\$A5A5A5A5,\$0FFC | | | | |
| BNE | RAMFAIL | BRANCH IF | | | |
| | | FAILURE | | | |
| MOVE.W | #\$3B00,\$450000 | LEDS: _ _ G Y _ | | | |
| BSR | ACTRAM | ACTIVATE ALL | | | |
| | | SYSTEM RAM | | | |
| MOVE.W | #\$3900,\$450000 | LEDS: _ Y G Y _ | | | |
| BSR | SINIT | INITIALIZE THE | | | |
| | | SERIAL PORT | | | |
| MOVE.W | #\$3000,\$450000 | LEDS: _ Y G _ _ | | | |
| MOVE.L | #HELLO,-(A7) | GET FIRST | | | |
| | | MESSAGE | | | |
| BSR | PRINTF | DISPLAY IT | | | |
| ADDQ.L | #4,A7 | ADJUST STACK | | | |
| MOVE.W | #\$3F00,\$450000 | LEDS: _ _ G _ _ | | | |
| CMP.L | #\$A5A5A5A5,\$FF8 | BEEN THROUGH | | | |
| | | THIS BEFORE? | | | |
| BEQ | WARMSIS | SKIP TEST IF SO | | | |
| BSR | TESTRAM | TEST ALL | | | |
| | | AVAILABLE RAM | | | |
| MOVE.L | 00,-(A7) | PUSH SIZE OF | | | |
| | | RAM | | | |
| MOVE.L | D0,-(A7) | PUSH IT AGAIN | | | |
| MOVE.L | #SIZE,-(A7) | PUSH MESSAGE | | | |
| BSR | PRINTF | DISPLAY IT | | | |
| ADD.L | #12,A7 | REPAIR STACK | | | |
| MOVE.L | #\$A5A5A5A5,\$FF8 | WRITE FLAG TO | | | |
| | | SHOW WE'VE BEEN | | | |
| | | THROUGH HERE | | | |
| | | ALREADY (SEE STARTUP) | | | |
| HELP\$: | | | | | |
| MOVE.L | #FIRST,-(A7) | GET SIGN-ON MESSAGE | | | |
| BSR | PRINTF | PRINT IT | | | |
| ADDQ.L | #4,A7 | FIX THE STACK | | | |
| WARMS1\$: | | | | | |
| MOVE.L | #MON_SP,A7 | ENTRY POINT THAT | | | |
| | | RESETS THE SP | | | |
| WARMS\$: | | | | | |
| BSR | DUMP_REG\$ | REINSTALL THE DUMP | | | |
| | | REGS VECTOR | | | |
| BSR | MONITOR\$ | BRANCH TO THE | | | |
| | | MONITOR WITHOUT | | | |
| | | RESETTING THE STACK | | | |
| | | POINTER. | | | |
| MOVE.L | #EXITMSG,-(A7) | TELL USER EXIT | | | |
| | | ISN'T GOING TO WORK | | | |
| BSR | PRINTF | PRINT IT | | | |
| ADDQ.L | #4,A7 | FIX THE STACK | | | |
| BRA | WARMS\$ | AN ENDLESS LOOP | | | |
| | | | MONITOR\$: | | |
| | | | LINK | A6,#BUFFER | SET THE TEXT |
| | | | | | BUFFER IN STACK SPACE |
| | | | MOVEM.L | 00-07/A0-A7,-(A7) | SAVE EVERYTHING |
| | | | LOOP: | | |
| | | | BSR | PROMPT | |
| | | | LEA | BUFFER(A6),A0 | POINT A0 AT |
| | | | | | TEXT STRING |
| | | | BSR | GETLINE | |
| | | | BSR | PROCESS\$ | |
| | | | BRA | LOOP | |
| * * NOTE: Exit from this routine is via the monitor EXIT command, * whose code follows in the case structure below and is labeled * EXIT\$. * * * PROCESS * * This is the core of the ROM monitor. It executes the command * found in the first two character positions of the lin pointed * at by A0. Note that this routine is available exter- nally via * the jump table. This permits a user to load a string with a * monitor command, put the string's address in A0 and JSR here * so the monitor can process the command. As is custom ary, any * string submitted to PROCESS must be terminated with a null byte. * Additionally, any monitor command must be two charac- ters long * and must begin in column one. * | | | | | |

| | | | | | |
|---------------------------|---------------------------|---------------------------------|-----------|---|-------------------|
| PROCESS: | | | BYTE | ' Available RAM: %D bytes. First non-RAM address: %X.' | |
| CMP.B | #0,(A0) | ANYTHING ON THE LINE? | | | |
| BEQ | PROCX | BRANCH IF NOT | BYTE | \$0D,\$0A | |
| MOVE.L | A0,A1 | YES, MOVE POINTER INTO A1 | BYTE | 0 | |
| BSR | UPPER | CONVERT TO UPPERCASE | PROC_TBL: | | |
| MOVE.W | (A1),D0 | GET THE FIRST TWO CHARACTERS | WORD | 8 | |
| MOVE.L | @PROC_TBL,A0 | GET ADDR OF PROCESS TABLE | BYTE | 'D','U' | DUMP COMMAND |
| BRA | CASE | DO THE COMMAND | LONG | DUMPMEM\$ | |
| | | | BYTE | 'S','W' | \$WEEP COMMAND |
| | | | LONG | SWEEP\$ | |
| PROCX: | | | BYTE | 'F','I' | FILL COMMAND |
| RTS | | RETURN TO MAIN LOOP | LONG | FILL\$ | |
| | | | BYTE | 'C','H' | CHANGE COMMAND |
| | | | LONG | CHANGES | |
| | | | BYTE | 'H','E' | HELP COMMAND |
| | | | LONG | HELPS | |
| DUMPMEM\$: | BSR DUMPMEM | | BYTE | 'R','L' | RAM LOAD |
| | BRA PROCX | | | | (S19) COMMAND |
| | | | LONG | S19\$ | |
| SWEEP\$: | BSR SWEEP | | BYTE | 'G','O' | GO COMMAND |
| | BRA PROCX | | LONG | GO\$ | |
| FILL\$: | BSR FILL | | BYTE | 'E','X' | EXIT COMMAND |
| | BRA PROCX | | LONG | EXITS | |
| CHANGES: | BSR CHANGE | | LONG | WHAT | DEFAULT CASE |
| | BRA PROCX | | | | |
| S19\$: | BSR S19 | HELLO: | BYTE | \$0D,\$0A,\$0A | |
| | BRA PROCX | | BYTE | ' 68010 ROM MONITOR V1.2' | |
| GO\$: | BSR GO | | BYTE | \$0D,\$0A | |
| | BRA PROCX | | BYTE | ' Written for the Convergent Technologies' | |
| EXITS: | | | BYTE | ' Mini-Frame' | |
| ADDQ.L #4,A7 | POP THE PROCESS RTN ADDR | | BYTE | \$0D,\$0A | |
| MOVEM.L (A7)+,D0-D7/A0-A7 | RESTORE EVERYTHING | | BYTE | 0 | |
| UNLK A6 | REMOVE THE FRAME | FIRST: | BYTE | \$0D,\$0A,\$0A | |
| RTS | LEAVE THE MONITOR ROUTINE | | BYTE | 'All commands are two characters, followed ' | |
| | | | BYTE | 'by any arguments.' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | 'Separate all arguments (in hex) by at least ' | |
| | | | BYTE | 'one space. Available' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | 'commands are:' | |
| | | | BYTE | \$0D,\$0A,\$0A | |
| | | | BYTE | ' Dump memory DU <addr>' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | ' Sweep memory SW <start> <stop> [times]' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | ' Change memory CH <addr> <data>...<data> or' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | ' CH <addr>' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | ' Fill memory FI <start> <stop> <data>' | |
| | | | BYTE | \$0D,\$0A | |
| | | | BYTE | ' RAM Load (S1-S9) RL <offset addr>' | |
| SIZE: | | | BYTE | \$0D,\$0A | |
| | | | | | |

```

BYTE      '      Goto location      GO
           <transfer addr>'
BYTE      $0D,$0A
BYTE      '      Exit monitor      EX'
BYTE      $0D,$0A,$0A
BYTE      0

```

END

```

EXT.W      D0
           * CLEAR HIGH
           BYTE
CMP.B      #'-',D0
BNE        SKOPF
           * IS IT MINUS?
           * NO, TRY
           * AGAIN
MOVE.W     #1,LEFTJ(A6)
           * SET LEFT
           * JUSTIFY FLAG
BRA        LPOPF
           * TRY FOR NEXT
           * CHAR

```

SKOPF:

```

CMP.B      #'0',D0
BLT        SKIPF
           * IS IT < 0?
           * YES,CONTINUE
           * PROCESSING
CMP.B      #'9',D0
BGT        SKIPF
           * IS IT > 9?
           * YES,CONTINUE
           * PROCESSING
MOVE.W     FIELD(A6),D1
           * GET CURRENT
           * FIELD SIZE
MULU      #10,D1
           * SHIFT LEFT
           * ONE DEC DIGIT
           * CONVERT
           * ASCII TO NUMBER
ADD.W      D0,D1
           * ADD TO FIELD
           * WIDTH
MOVE.W     D1,FIELD(A6)
BRA        LPOPF
           * SAVE IT
           * GET NEXT
           * FORMAT CHAR

```

SKIPF:

```

MOVE.L     #DISPATCH,A0
           * GET CASE
           * TABLE ADDR
BRA        CASE
           * DO CASE

```

NO_CTL:

```

MOVE.W     D0,-(A7)
BSR        PUTC
           * PUSH CHAR
           * PRINT IT
ADDQ.L     #2,A7
           * TRASH
           * PARAMETER
           * DO IT AGAIN
BRA        LOOP

```

EXIT:

```

MOVEM.L    (A7)+,D0-D6/A0-A2
UNLK       A6
RTS

```

*

*

D_ARG:

```

MOVE.W     (A2)+,D0
           * GET VALUE,
           * MOVE POINTER
EXT.L      D0
           * CONVERT TO
           * COMMON FORMAT
BSR        SIGN
           * PRINT SIGN,
           * GET ABS()
BRA        PRINTDEC
           * PRINT VALUE

```

.

*

U_ARG:

```

MOVE.W     (A2)+,D0
           * GET VALUE,
           * MOVE POINTER
AND.L      #$0000FFFF,D0
           * ZERO HIGH
           * WORD
BRA        PRINTDEC
           * PRINT VALUE

```

*

*

D1_ARG:

```

MOVE.L     (A2)+,D0
           * GET VALUE,
           * MOVE POINTER
BSR        SIGN
           * PRINT SIGN,
           * GET ABS()
BRA        PRINTDEC
           * PRINT VALUE

```

*

*

U1_ARG:

```

MOVE.L     (A2)+,D0
           * GET VALUE,

```

TITLE PRINTF.ASM

```

*
* PRINTF - Subset/superset of C printf standard I/O
function
*
* Control args:
*   push last parameter in control string first, then
next-to-
*   last, etc. Push addr of control string last.
*
* %d Print signed decimal word
* %u Print unsigned decimal word
* %D Print signed decimal longword
* %U Print unsigned decimal longword
* %x Print hexadecimal word
* %X Print hexadecimal longword
* %s Print null-terminated string
* %c Print character
* %v Cursor (x,y) - push x, then y as words
* %default Print next character as literal
*
* Taken from '68000 Assembly Language,' by Krantz and
Stanley.
* Listing appears on page 234.
*

```

```

PUBLIC     PRINTF$
EXTERN     CASE,PUTC,CURSOR

```

```

* LOCAL VARIABLE DISPLACEMENT DEFINITIONS
*

```

```

LEFTJ     EQU      -2
FIELD     EQU      -4
SIGNF     EQU      -6

```

PRINTF\$:

```

LINK      A6,#-6
MOVEM.L    D0-D6/A0-A2,-(A7)
MOVE.L     8(A6),A1
           * GET CONTROL
           * STRING ADDR
LEA        12(A6),A2
           * GET POINTER
           * TO PARAMETERS

```

LOOP:

```

MOVE.B     (A1)+,D0
           * GET CONTROL
           * STRING CHAR
BEQ        EXIT
           * QUIT IF IT'S
           * A NULL
EXT.W      D0
           * CLEAR HIGH
           * BYTE
CMP.B      #'%',D0
           * SEE IF IT'S
           * CONTROL FLAG
BNE        NO_CTL
           * BRANCH IF
           * NOT
CLR.W      LEFTJ(A6)
           * CLEAR LEFT
           * JUSTIFY FLAG
CLR.W      FIELD(A6)
           * CLEAR FIELD
           * WIDTH
CLR.W      SIGNF(A6)
           * CLEAR SIGN
           * FLAG

```

LPOPF:

```

MOVE.B     (A1)+,D0
           * GET NEXT

```

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809"

which requires the TSC Macro Assembler.

FLEX, SK-DOS, CCF - \$99.95

Macro Assembler for TSC - The FLEX, SK-DOS STANDARD Assembler.

Special -- CCF \$35.00; FLEX, SK-DOS \$50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF: Generate OS-9 Memory modules under FLEX, SK-DOS.

FLEX, SK-DOS, CCF, OS-9 \$99.00

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.

FLEX, SK-DOS, CCF \$150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler. fast interactive A.L. Programming for small to medium-sized Programs.

FLEX, SK-DOS, CCF - \$75.00

XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8

FLEX, SK-DOS, CCF - \$98.00

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive

Disassembler; extremely POWERFUL! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF: Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/ A.L. Source)

CCD (32K Req'd) Object Only \$49.00

FLEX, SK-DOS \$99.00 - CCF Object Only \$50.00 UniFLEX \$100.00

CCF, with Source \$99.00 OS-9, \$101.00 - CCO, Object Only \$50.00

68010 SUPER SLEUTH - Similar to 8-Bit Version except written in "C".

68010 Disassembler \$100.00 FLEX, UniFLEX, UNIX, XENIX,

MS-DOS, SK-DOS, OS-9

OS-9/68K Object Only \$100.00 or with Source \$200.00

DYNAMITE+ -- Excellent standard "Hatch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options with OS-9 Version.

CCF, Object Only \$100.00 - CCO, Object Only \$59.95

FLEX, SK-DOS, Object Only \$100.00 - OS-9, Object Only \$150.00

UniFLEX Object Only \$300.00

CROSS ASSEMBLERS

CROSS ASSEMBLERS from Computer System Consultants -- Supports

1802/5, Z-80, 6800/1/2/3/8/11/11C11, 6804, 6805/11C05/ 146805, 6809/00Q01, 6502 family, 8080/5, 8020/1/2/3/5/C35/39/ 40/48/C48/49/C49/50/ 8748/49, 8031/51/8751, 32000 and 6800Q/68010 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text. Includes Macro Pre-Processor. Written in "C". 68000 or 6809 *Macintosh *Atari, FLEX, CCF, UniFLEX, OS-9, XENIX, UNIX, MS-DOS, SK-DOS

any object \$50 or any 3 for \$100

any source is an additional \$50 or any 3 for \$100

Set of ALL object \$200.00 - with source \$500.00

XASM Cross Assemblers for FLEX, SK-DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for target CPU's.

Complete set, FLEX, SK-DOS only - \$150.00

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS-9 (binary), and FLEX, SK-DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

| For: | MOTOROLA | INTEL | OTHER | COMPLETE SET |
|-----------|----------|-------|-------|--------------|
| FLEX9 | \$150 | \$150 | \$150 | \$399 |
| SK-DOS | \$150 | \$150 | \$150 | \$399 |
| OS-9/6809 | \$150 | \$150 | \$150 | \$399 |
| OS-9/68K | ***** | ***** | ***** | \$432 |

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS-9/68K on your OS-9/6809 computer.

FLEX, SK-DOS, CCF, OS-9/6809 \$249.00

COMMUNICATIONS

C-MODEM Telecommunications Program from Computer Systems

Consultants, Inc. -- Menu-Driven: supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, SK-DOS, CCF, OS-9, UniFLEX, UNIX, XENIX, MS-DOS, with Source \$100.00 - without Source \$50.00

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gmix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) \$99.95

X-TALK Software (2 disks only) \$69.95

X-TALK with C-MODEM Source \$149.95

XIDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC: Re-Transmits bad blocks, etc.

UniFLEX - \$299.99

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. \$500+ page Manual with tutorial guide.

FLEX, SK-DOS, CCF - \$198.00

PASC from S.E. Media - A FLEX9, SK-DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.

FLEX, SK-DOS \$95.00

WHIMSICAL from S.E. MEDIA Now supports Real Numbers. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

FLEX, SK-DOS and CCF - \$195.00

KANSAS CITY BASIC from S.E. Media - Basic for Color Computer OS-9 with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT\$, RIGHT\$, MID\$, STRING\$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 \$39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK-DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

FLEX, SK-DOS, CCF - \$295.00

C Compiler from Intel -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler, FAST, efficient Code. More UNIX Compatible than most.

FLEX, SK-DOS, CCF, OS-9 (Level II ONLY), UniFLEX - \$575.00

PASCAL Compiler from Luclata -- ISO Based P-Code Compiler.

Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

FLEX, SK-DOS and CCF - \$190.00

OmegaSoft PASCAL from Certified Software -- Extended Pascal for systems and real-time programming.

Native 68000/68020 Compiler, \$575 for base package, options available. For OS-9/68000 and PDOS host system.

6809 Cross Compiler (OS-9/68000 host) \$700 for complete package.

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

FLEX, SK-DOS, CCF, OS-9 Compiler /Assembler \$99.00

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSI Level 1 COBOL with many of the useful Level 2 features. Full FLEX, SK-DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. A very popular product.

FLEX, SK-DOS, CCF - \$99.95

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - \$58.95

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.

FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.

Like compilers for other languages, FORTHBUILDER can operate in "batch mode".

The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.

FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.

FLEX, CCF, SK-DOS - \$99.95

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Grafix); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

* Now supplied as a two disk set:

Disk #1: JUST2.CMD object file,

JUST2.TXT PL9 source: FLEX, SK-DOS - CCF

Disk #2: JUSTSC object and source in C:

FLEX, SK-DOS, OS-9, CCF

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN, 37343



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola.*FLEX and UniFLEX are Trademarks of Technical Systems Consultants.*SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

source compiles to a standard syntax JUST.COM object file. Using JUST syntax (.p, .u, .y, etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - *PLY FLEX only: FLEX, SK-DOS & CCF* - \$49.95

Disk Set (2) - *FLEX, SK-DOS & CCF & OS-9 (C version)* - \$69.95

OS-9 68K000 complete with Source - \$79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX, SK-DOS \$129.50

* SPECIAL INTRODUCTION OFFER * \$79.95

SPECIAL PAT/JUST COMBO (with source)

FLEX, SK-DOS \$99.95

OS-9 68K Version \$229.00

SPECIAL PAT/JUST COMBO 68K \$249.00

Note: JUST in "C" source available for OS-9

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassle' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - approx. 14,000 plus of free memory! Extra fine for programming as well as text.

FLEX, SK-DOS \$69.95

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, SK-DOS \$39.95

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX, SK-DOS or SSB-DOS, OS-9 - \$175.00

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words!

Look up a word from within your Editor or Word Processor (with the SPH CMD Utility which operates in the FLEX, SK-DOS UCS). Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

FLEX, SK-DOS and CCF - \$129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - \$99.95,

FLEX, SK-DOS or OS-9 - \$179.95, UniFLEX - \$299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - \$69.95,

FLEX, SK-DOS or OS-9 - \$99.95, UniFLEX - \$149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - \$59.95,

FLEX, SK-DOS or OS-9 - \$79.95, UniFLEX - \$129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!

FLEX, SK-DOS or OS-9 - \$329.95, UniFLEX - \$549.95

OS-9 68000 \$695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems

FOR 6809 FLEX or SK-DOS (512K)

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Whiten in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

Availability Legend

O = OS-9, S = SK-DOS

F = FLEX, U = UniFLEX

CCO = Color Computer OS-9

CCF = Color Computer FLEX



South East Media

5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **

Add 1% U.S.A. (min. \$2.50)

Foreign Surface Add 5%

Foreign Airmail Add 10%

Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola.*FLEX and UniFLEX are Trademarks of Technical Systems Consultants.*SK-DOS is a Trademark of Star-K Software Systems Corp.



PC-68K1

PC Co-processor

ULTRASCIENCE

Easily installed, the PC-68K1 expands personal computers with high-performance, versatile, 16 bit, 68K CPU processing power. DOS runs concurrently with real-time, multi-user, multi-tasking OS-9.

© Copyright Ultrascience 1988



The PC-68K1 PC Co-Processor

Personal Computers Become Computer Systems

Why?

The standardized, widely accepted, and ubiquitous IBM Personal Computer represents an enormous, yet largely untapped, resource of computing power. Unless connected to a costly LAN and fileserver, the single-user "PC" provides neither the level of interaction nor the convenient sharing of data so necessary in business, science, or industry. However, the simple addition of a PC-68K1 expands and enhances a PC into a real-time, multi-user, multi-tasking, professional computer system, running OS-9 and MS-DOS concurrently. A PC-68K1/OS-9 equipped PC can host a wide variety of high performance application software, previously reserved for much larger and far more expensive machines, without sacrificing MS-DOS applications. The OS-9 operating system offers extensive networking capability and a range of popular languages, including C, BASIC, PASCAL, FORTRAN, etc. Powerful C language VAX/UNIX extensions and a special set of OS-9/UNIX library routines make converting UNIX applications to the more efficient OS-9 a simple and straightforward task.

How?

All that is required to get on-line is an operating PC, a PC-68K1 and OS-9. The PC-68K1 allows the user to flip between active MS-DOS and OS-9 tasks, using the Personal Computer monitor, keyboard, and existing disk drives. MS-DOS files are available to OS-9 and vice versa.

Now...

Even PC-68K1 equipped PCs, used exclusively for software development or education, do not remain single-user for long; that would be a terrible waste of power. Terminals and all manner of peripherals and interfaces can be connected to the PC-68K1 ports. Substantial caching disk drives, high-speed tape backup, and special purpose boards can be added to the PC bus. In short, the Personal Computer becomes a truly professional, cost-effective system that means productivity.

Specifications

Function

- Co-processor for IBM PC/XT/AT and compatibles.

Operating Systems

- Real-time, Multi-user, Multi-tasking, OS-9 executes concurrently with MS-DOS.

Microprocessor

- 68000/68010 MHz and 10/12.5 MHz.

Real-Time Clock

- Time of day and date clock with battery back-up.

Memory

- 1 M bytes of dynamic RAM running with 0 wait states.
Note: Dedicated 68K bus 0 wait states dynamic RAM expansion boards are available.
- 32 K bytes of EPROM.
- 8 K bytes of battery backed up static RAM (optional).
- Memory mapped, dual ported, 1K RAM interface supports high-speed DMA transfers to and from PC bus.

Serial Ports

- Two standard DB9S ports, asynchronous selectable to 19.2 baud or synchronous.
Note: Dedicated 68K bus 10 port serial expansion boards are available.

Parallel Ports

- One Centronics port or two 8 bit parallel ports.

Diagnostic Capabilities

- An automatic self-test is performed at OS-9 boot-up.
- Sophisticated software utilities are also provided.

Reset

- Bus-independent, software or switch controlled, hardware reset.

Physical

- 13.2" x 4.2" standard, full length 8 bit bus slot.

Electrical

- 5V @ 1.3A, +12 @ 100 mA, -12 @ 100 mA.

Operating Environment

- 5° C to 55° C and maximum 90% relative humidity at board surface.

Software Support

- An extensive list of drivers and utilities is available.



ULTRASCIENCE

DIVISION OF

GIBBS LABORATORIES, INC.

COMPUTER

SYSTEMS

INTEGRATION

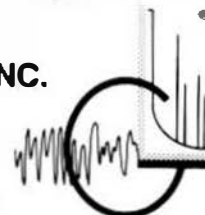
FAX # 1-312-256-0097

PHONE # 1-312-256-0080

TELEX # 910-997-0379

1824 WILMETTE AVE. • P.O. BOX 558 • WILMETTE, IL 60091

©Copyright 1988



Trademarks: S/R and PC-68K1 - ULTRASCIENCE; OS-9 - Microware Systems Corp.; MS-DOS - Microsoft Corp.; IBM - IBM Corp.; UNIX - AT&T Bell Laboratories; VAX - Digital Equipment Corp.
Specifications subject to change without notice.

SR[™]

MEMIOX

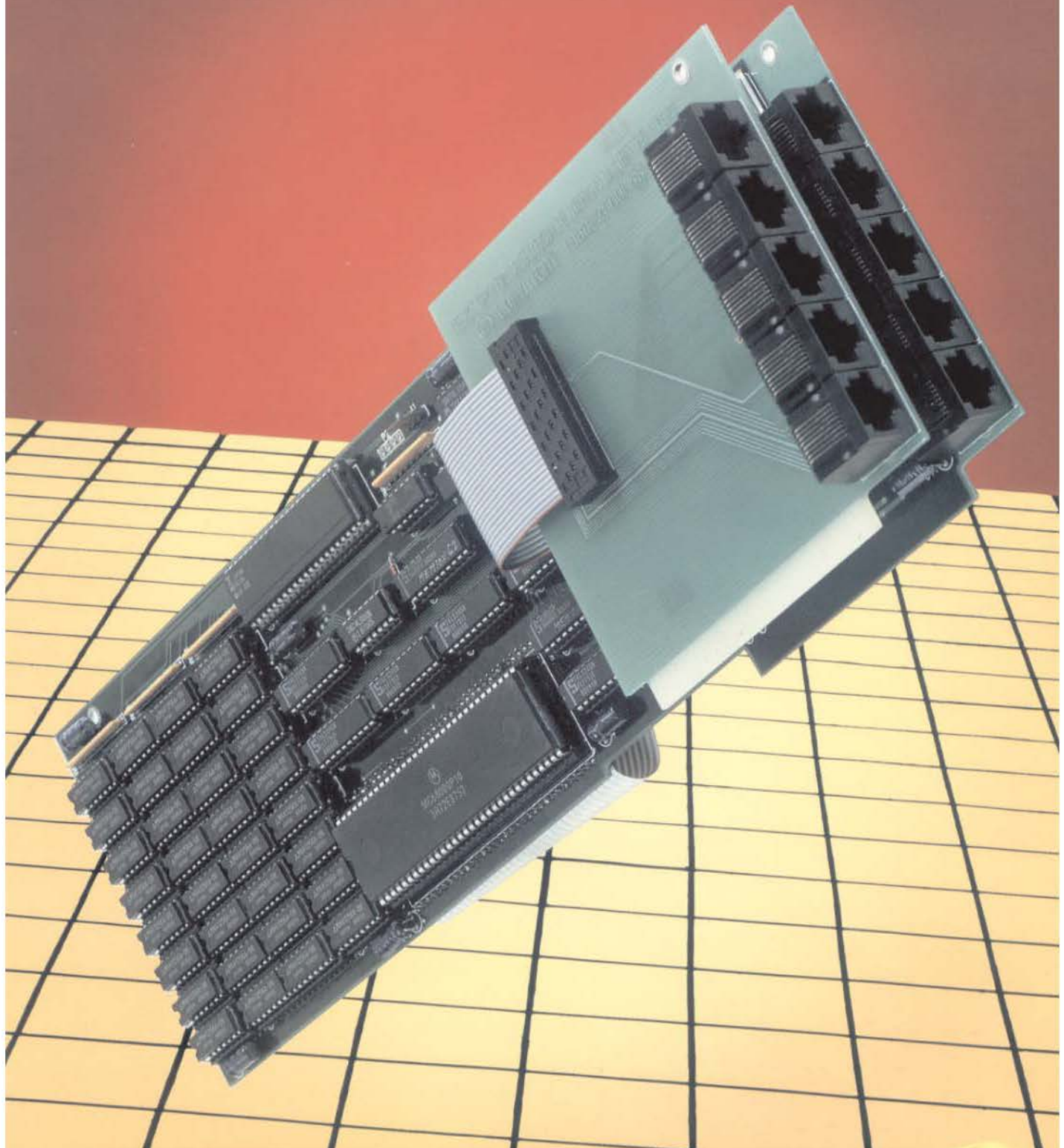
Memory / IO

Expansion

ULTRASCIENCE

Easily installed, the MEMIOX expands PC-68K1 equipped personal computers by 1 MB of zero wait states RAM and 10 real-time, multi-user, multi-tasking, serial ports.

© Copyright Ultrascience 1988



The MEMIOX Memory/IO Expansion Board

Expand Your PC to 14 Ports and 2 Megs RAM

Why?

Anyone who has discovered the capability of real-time, multi-user, multi-tasking OS-9 and the power of the PC-68K1 co-processor knows that adding 10 more efficient serial ports and another megabyte of 0 wait state RAM to a PC-68K1 will turn a "PC" into a substantial computer system. Such a system can run a business, control a plant, automate a laboratory, or support an entire class, and the price is right!

How?

The MEMIOX board plugs onto the PC-68K1 and it is ready to go. For standard installations, there are no jumpers or switches to set. The convenient RJ45 port connectors are mounted right on the MEMIOX board set, so there is no clumsy hardware to attach and connecting peripherals could not be easier. Ultrascience offers a complete line of shielded RJ45 modular cable and adapters to make cabling a snap.

Now...

The expanded PC-68K1 system is based on standard, large-scale, open-architecture design, which translates to smooth, predictable, upgrade paths for both hardware and software well into the future.

Specifications

Function

- Expands the RAM and/or I/O of an IBM PC/XT/AT compatible PC-68K1 co-processor, using the high-speed 68K bus.

Memory

- 1 M bytes of dynamic RAM running with 0 wait states.

Serial Ports

- Ten standard RJ45 ports, selectable to 19.2 K baud.

Diagnostic Capabilities

- An extensive automatic self-test is performed at OS-9 boot-up. Sophisticated software utilities are also provided.

Physical

- 13.2" X 4.2" standard, full-length 8 bit bus slot and a slot for mechanical support of the daughter board.

Electrical

- 5v @ 1.3A, +12 @ 500 mA, -12 @ 500 mA.

Operating Environment

- 5°C to 55°C and maximum 90% relative humidity at board surface.

Software Support

- Memory and Serial Ports supported on OS-9.



ULTRASCIENCE

COMPUTER

FAX # 1-312-256-0097

DIVISION OF

SYSTEMS

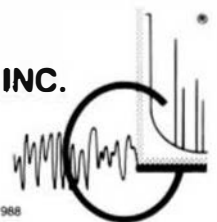
PHONE # 1-312-256-0080

GIBBS LABORATORIES, INC.

INTEGRATION

TELEX # 910-997-0379

1824 WILMETTE AVE. ■ P.O. BOX 558 ■ WILMETTE, IL 60091

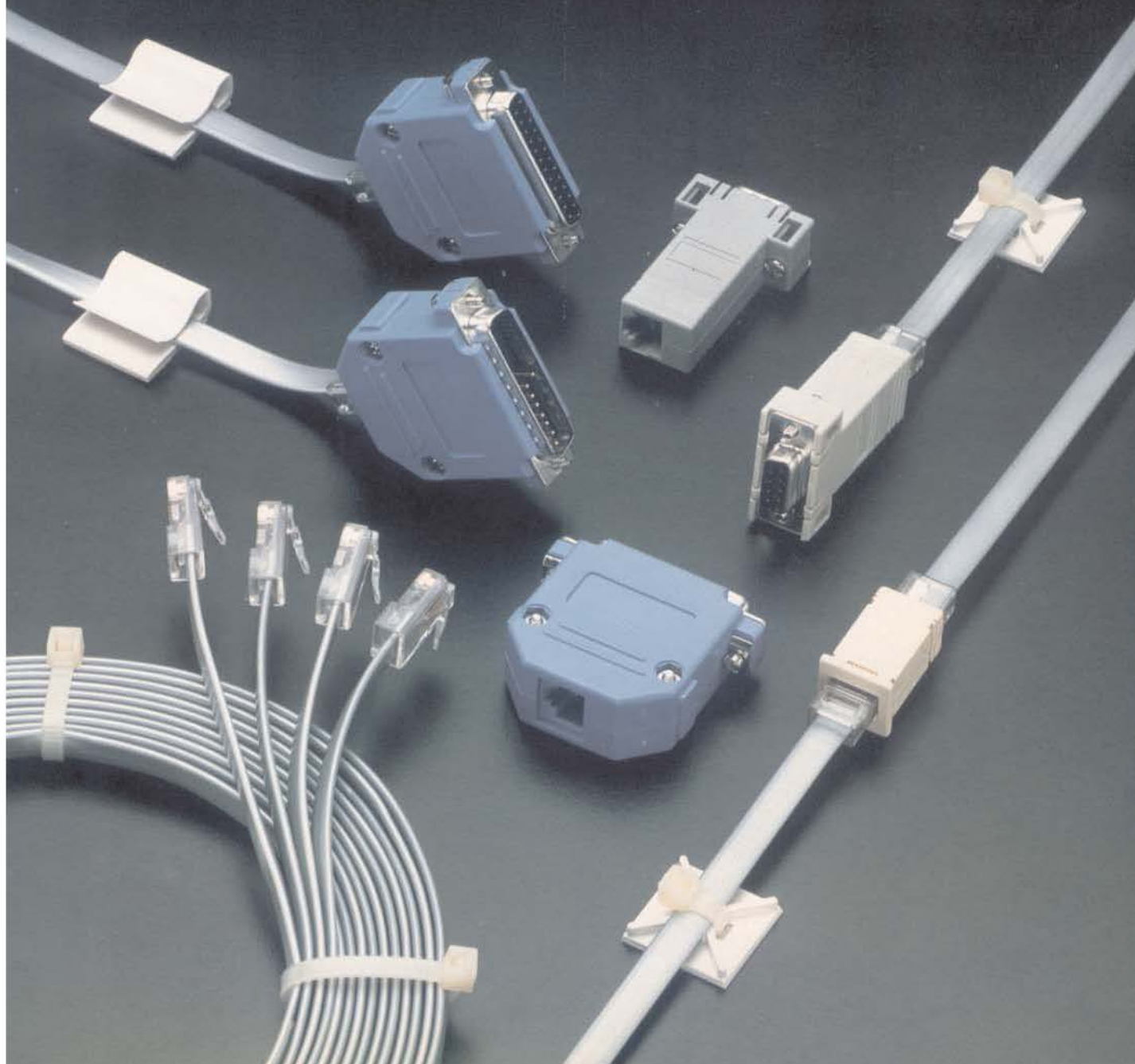


©Copyright 1988

SR PC68K1 & SR MEMIOX are trademarks of ULTRASCIENCE. OS-9 is a trademark of Microware. Specifications subject to change without notice.

ULTRASCIENCE

© Copyright Ultrascience 1988



"CLICK"

RJ45 Cables and Connectors

Cables

Shielded, universal "CLICK" Cables are resistant to flame, oils, and many kinds of typical mechanical abuse,¹ and they are ready for use without tools. Eight low resistance,² fully annealed, copper conductors provide enough circuits for essentially all computer peripherals. High quality insulation and low inter-lead capacitance³ make the use of long lengths practical. A 100% coverage aluminum/polyester foil shield controls electromagnetic interference (EMI). The quick-release, latching RJ45 connectors are built with gold plated contacts for reliable, low noise, low loss connections. The small cross section of the connectors permits them to be threaded through tiny openings.

¹ UL style 20251 ² Typ. 100 ohm/ftm ³ Typ. 32 pF/m

Connectors

Changing a device is a simple "CLICK"! It is the "CLICK" Connectors which bring method to the madness of cabling computers. A few standard styles and genders can be mixed and matched at either end of the universal "CLICK" Cables in order to interconnect terminals, printers, modems, PC COM ports etc. "CLICK" Connectors have gold plated contacts and high-impact, flame resistant, oil resistant plastic bodies.

Routing Aids

"CLICK" Routing Aids make it easy to keep cables from getting tangled, caught, or pinched. Cables can be bundled or attached to a surface in a jiffy.

Part Numbers

Cables

- | | |
|------------|---------------------|
| • SH-RJ-3 | 3 Foot (~1 Meter) |
| • SH-RJ-10 | 10 Foot (~3 Meters) |
| • SH-RJ-20 | 20 Foot (~6 Meters) |
| • SH-RJ-30 | 30 Foot (~9 Meters) |
| • SH-RJ-X | Custom length |

Connectors

- | | |
|----------------|---------------------------------------|
| • RJ-RJ-STD | RJ45 by RJ45 — couples/extends cables |
| • FDB25-RJ-STD | Female DB25 — terminals/printers |
| • MDB25-RJ-STD | Male DB25 — terminals/printers |
| • FDB25-RJ-NM | Female DB25 — null modem |
| • MDB25-RJ-NM | Male DB25 — null modem |
| • FDB25-RJ-NC | Female DB25 — custom |
| • MDB25-RJ-NC | Male DB25 — custom |
| • MDB9-RJ-K1S | Male DB9 — S/R PC-68K1 serial port |
| • MDB9-RJ-PCS | Female DB9 — PC COM serial port |
| • MDB9-RJ-NC | Male DB9 — custom |
| • FDB9-RJ-NC | Female DB9 — custom |

Routing Aids

- | | |
|------------|---|
| • RA-MOUNT | Adhesive-backed clip for surface mounting cables |
| • RA-CLAMP | Adhesive-backed surface mount for use with cable ties |
| • RA-TIE | Cable ties |



ULTRACIENCE

DIVISION OF

GIBBS LABORATORIES, INC.

COMPUTER

SYSTEMS

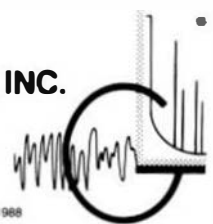
INTEGRATION

FAX # 1-312-256-0097

PHONE # 1-312-256-0080

TELEX # 910-997-0379

1824 WILMETTE AVE. • P.O. BOX 558 • WILMETTE, IL 60091



©Copyright 1988

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

FOR 6809 FLEX or SK-DOS(5"/8" Disk) **\$249.95**

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

OS-9 & CCO object only -- \$39.95; with Source -- \$79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - for your programs - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

OS-9 & CCO object only - \$89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

FLEX, SK-DOS, CCF -- EACH 5" - \$40.00, 8" - \$50.00

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works with ALL Versions of 6809 UniFLEX basic.

UniFLEX - \$219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK-DOS on either 5" or 8" Disk.

1. BASIC TOOL-CHEST \$29.95

BLISTER.CMD: pretty printer

LNEXREF.BAS: line cross-referencer

REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code

STRIP.BAS: superfluous line-numbers stripper

2. FLEX, SK-DOS UTILITIES KIT \$39.99

CATS. CMD: alphabetically-sorted directory listing

CATD.CMD: date-sorted directory listing

COPYSORT.CMD: file copy, alphabetically

COPYDATE.CMD: file copy, by date-order

FILEDATE.CMD: change file creation date

INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents

RELINK.CMD (& RELINK82): re-orders fragmented free chain

RESQ.CMD: undeletes (recovers) a deleted file

SECTORS.CMD: show sector order in free chain

XL.CMD: super text lister

3. ASSEMBLERS/DISASSEMBLERS UTILITIES \$39.95

LINEFEED.CMD: 'modularise' disassembler output

MATH.CMD: decimal, hex, binary, octal conversions & tables

SKIP.CMD: column stripper

4. WORD - PROCESSOR SUPPORT UTILITIES \$49.95

FULLSTOP.CMD: checks for capitalization

BSTYCI.BAS (.BAC): Stylo to dot-matrix printer

NECPRI.CMD: Stylo to dot-matrix printer filter code

5. UTILITIES FOR INDEXING \$49.95

MENU.BAS: selects required program from list below

INDEX.BAC: word index

PHRASES.BAC: phrase index

CONTENT.BAC: table of contents

INDXSORT.BAC: fast alphabetic sort routine

FORMATER.BAC: produces a 2-column formatted index

APPEND.BAC: append any number of files

CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09.

6 were written in C Language and the remainder in assembly.

All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters

2. DPEEK -- Double peck

3. DPOKE -- Double poke

4. FPOS -- Current file position

5. FSIZE -- File size

6. ITRIM -- removes leading spaces from a string

7. GETPR -- returns the current process ID

8. GETOPT -- gets 32 byte option section

9. GETUSR -- gets the user ID

10. GTIME -- gets the time

11. INSERT -- insert a string into another

12. LOWER -- converts a string into lowercase

13. READY -- Checks for available input

14. SETPRIOR -- changes a process priority

15. SETUSR -- changes the user ID

16. SETOPT -- set 32 byte option packet

17. STIME -- sets the time

18. SPACE -- adds spaces to a string

19. SWAP -- swaps any two variables

20. SYSCALL -- system call

21. UPPER -- converts a string to uppercase

For OS-9 - \$44.95 - Includes Source Code

SOFTTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK-DOS files.

COMMON compare two text files and reports differences.

COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK-DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legend:

● = OS-9, S = SK-DOS

F = FLEX, U = UniFLEX

CC = Color Computer OS-9

CT = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **

Add 2% U.S.A. (min. \$2.50)

Foreign Surface Add 5%

Foreign Airmail Add 10%
Or C.G.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

ECHO echoes to either screen or file.

FIND an improved find command with "pattern" matching and wildcards.

Very useful.

HEX dumps files in both hex and ASCII.

INCLUDE a file copy program that will accept "includes" of other disk files.

KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.

LISTDIR a directory listing program. Not super, but better than CAT.

MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted.

Very fast. Very useful.

MULTICOL width of page, number of columns may be specified. A MUST!

PAGE similar to LIST but allows for a page header, page width and depth.

Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.

REMOVE a fast file deleter. Careful, no prompts issued. Zap, and it's gone!

SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.

SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on *n*th word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.

TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.

TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.

UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.

WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, with source (PL9). 3 5-1/4" disks or 1 8" disk without source.

Complete set SPECIAL INTRO PRICE:

5-1/4" with source FLEX or SK-DOS - \$129.95

without source - \$79.95

8" with source - \$79.95 - without source \$49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -

TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.

FLEX, SK-DOS and CCF, UniFLEX - \$25.00, with Source - \$50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!

Levels I & II only - OS-9 \$69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.

Level I OS-9 object \$79.95; with Source \$149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes:

REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK-DOS Format so it can be used normally by FLEX, SK-DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK-DOS Directory, Delete FLEX, SK-DOS Files, Copy both directions, etc. FLEX, SK-DOS users use the special disk just like any other FLEX, SK-DOS disk

OS-9 - 6809/68000 \$79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.

OS-9 \$85.00

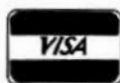
HIER from S.E. Media - HIER is a modern hierarchical storage system for users under FLEX, SK-DOS. It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK-DOS disk (8" 5" hard disk) can have sub directories. By this method the problems of assigning unique names to files is less cumbersome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. Each directory looks to FLEX, SK-DOS like a regular file, except they have the extension '.DIR'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK-DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

FLEX - SK-DOS \$79.95

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK-DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fouting with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK-DOS, 8" or 5") \$99.50

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC = Color Computer OS-9
CCP = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



•• Shipping ••
Add 1% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

OS-9 is a Trademark of Microware and Motorola. FLEX and UniFLEX are Trademarks of Technical Systems Consultants. SK-DOS is a Trademark of Borland Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

Fax: (615) 842-7990

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK-DOS, SSB-DOS68, and Digital Research CP/M Disks while operating under SK-DOS, FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

FLEX, SK-DOS and CCF 5" - \$50.00 FLEX, SK-DOS 8" - \$65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL*, and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.

6809 OS-9 & CCO - object only - \$49.95

FLEX, SK-DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (with Source Code) FLEX, SK-DOS Utilities for every FLEX, SK-DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten X BASIC Programs including: A BASIC Resequencer with EXTRAS over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, X BASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or A.L. Source Code).

FLEX, SK-DOS and CCF - \$50.00

BASIC Utilities ONLY for UniFLEX -- \$30.00

MS-DOS to FLEX Transfer Utilities to OS-9 For 68XXX and CCoS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

**CoCo Version: \$69.95*

68XXX Version \$99.95

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.

*UniFLEX - \$395.00. FLEX, SK-DOS, OS-9 and SPECIAL CCF - \$250.00
OS-9 68K - \$299.00*

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DIET-TRAC Forecaster from S.E. Media -- An X BASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

FLEX, SK-DOS - \$59.95, UniFLEX - \$89.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK-DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels)

FLEX, SK-DOS and CCF - \$79.95

NEW

MS-DOS/FLEX Transfer Utilities For 68XXX and CoCo OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.

CCO \$69.95 68XXX OS-9 \$99.95

MS-DOS and Macintosh Software at Discounted Prices

"Call for prices, it'll be worth the savings."

(615) 842-4600

FAX (615) 842-7990

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

| BRA PAGE | PRINTDEC | MOVE POINTER * PRINT VALUE | LPD_PR: | SPACES NEEDED |
|---|-------------------------|-------------------------------|--|--------------------------------|
| | | | TO _PUT | SAVE D0 ACROSS CAL |
| * SIGN | | | MOVE.W D0, -(A7) | |
| * PRINT SIGN IF NEEDED AND TAKE ABS() OF VALUE. | | | MOVE.W #02020, -(A7) | 1 SPACE TO OUTPUT |
| SIGN: | TST.L D0 | * IS IT NEGATIVE? | BSR PUTC | SEND THE SPACE |
| | BPL SK0_SG | * EXIT IF NOT | ADDQ.L #2, A7 | ADJUST STACK |
| | MOVE.W #0-1, SIGNF(A6) | * FLAG SIGN NEEDED | MOVE.W (A7)+, D0 | RETRIEVE D0 |
| | SUBQ.W #1, FIELD(A6) | * TAKE AWAY ONE FOR SIGN | SUBQ.W #1, FIELD(A6) | DECREMENT LOOP INDEX |
| | NEG.L D0 | * MAKE ABS() | BNE LPD_PR | LOOP IF MORE SPACES |
| SK0_SG: | RTS | | CHKSIGN: | |
| | | | TST.W SIGNF(A6) | SIGN NEEDED? |
| | | | BEQ CHKEXIT | JUMP IF NOT |
| | | | MOVE.L D0, D2 | SAVE D0 ACROSS CAL |
| | | | TO _PUT | |
| | | | MOVE.W #0202D, -(A7) | PUSH SIGN |
| | | | BSR PUTC | SEND IT |
| | | | ADDQ.L #2, A7 | ADJUST STACK |
| | | | MOVE.L D2, D0 | RETRIEVE D0 |
| | | | CHKEXIT: | |
| | | | RTS | |
| | | | PAGE | |
| * PRINTDEC | | | | |
| * COMMON DECIMAL OUTPUT ROUTINE. VALUE IS IN D0 UPON ENTRY. | | | * POSTFIX | |
| | | | * PRINTS ANY POSTFIX SPACES. | |
| PRINTDEC: | | | | |
| | CLR.W D1 | * OUTPUT DIGIT COUNT | POSTFIX: | |
| LP0_PD: | | | SUB.W D6, FIELD(A6) | DIGITS ALLOWED - ACTUAL |
| | DIVU #10, D0 | * DIVIDE NUMBER BY 10 | | EXIT IF NOT NEEDED |
| | BVS O_FLOW | * NUMBER TOO LARGE | BLE SK1_PO | |
| | SWAP D0 | * GET REMAINDER IN D0.W | | |
| | MOVE.W D0, -(A7) | * PUSH DIGIT | LP0_PO: | |
| | ADDQ.W #1, D1 | * BUMP DIGIT COUNT | MOVE.W #02020, -(A7) | SPACE TO SEND OUTPUT THE SPACE |
| | CLR.W D0 | * GET RID OF REMAINDER | BSR PUTC | ADJUST STACK COUNT THIS SPACE |
| | SWAP D0 | * PUT QUOTIENT IN D0.W | ADDQ.L #2, A7 | LOOP UNTIL DONE |
| | TST.W D0 | * IF ZERO, ALL DONE | SUBQ.W #1, FIELD(A6) | |
| | BNE LP0_PD | * LOOP IF NOT DONE | BNE LP0_PO | |
| | MOVE.W D1, D6 | * USED FOR FIELD ADJUST | | |
| | BSR PREFIX | * DO PRFIX SPACES | | |
| | SUBQ.W #1, D1 | * ADJUST LOOP INDEX CNTR | | |
| LP2_PD: | | | SK1_PO: | |
| | ADD.W #030, (A7) | * MAKE DIGIT ON TOS -> ASCII | RTS | |
| | BSR PUTC | * SEND TO OUTPUT | PAGE | |
| | ADDQ.L #2, A7 | * EAT DIGIT FROM TOS | X_ARG: | |
| | DBF D1, LP2_PD | * LOOP UNTIL ALL DONE | MOVE.W #3, D1 | NUMBER OF DIGITS TO PRINT |
| | BSR POSTFIX | * DO POSTFIX SPACES | MOVE.W #4, D6 | USED FOR FIELD ADJUST |
| | BRA LOOP | * EXIT TO CONTROL PARSER | MOVE.W (A2)+, D2 | TRANSFER OUTPUT VALUE POSITION |
| O_FLOW: | | | SWAP D2 | OUTPUT VALUE DO IT |
| | MOVE.L #OFLOWSTR, -(A7) | * PUSH CONTROL STRING ADDR | BRA PRINTEX | |
| | BSR PRINTFS | * PRINT IT | | |
| | ADDQ.L #4, A7 | * ADJUST STACK | | |
| | BRA LOOP | * CONTINUE | | |
| OFLOWSTR: | | | X1_ARG: | |
| | DB "overflow", 0 | | MOVE.W #7, D1 | NUMBER OF DIGITS TO PRINT |
| | PAGE | | MOVE.W #8, D6 | USED FOR FIELD ADJUST |
| * PREFIX | | | MOVE.L (A2)+, D2 | TRANSFER OUTPUT VALUE |
| * OUTPUT ANY NEEDED PREFIX SPACES AND SIGN. | | | BRA PRINTEX | GO PRINT IT |
| | | | PAGE | |
| PREFIX: | | | | |
| | TST.W FIELD(A6) | CHECK IF FIELD NONZERO | | |
| | BLE CHKSIGN | IF ZERO, SKIP NEXT PART | * PRINTEX | |
| | TST.W LEFTJ(A6) | LEFT JUSTIFY SELECTED? | | |
| | BNE CHKSIGN | BRANCH IF NOT | * OUTPUTS VALUE IN D2 IN HEX. D1 IS NUMBER OF DIGITS TO PRINT. | |
| | SUB.W D6, FIELD(A6) | DIGITS ALLOWED - ACTUAL | | |
| | BLE CHKSIGN | BRANCH IF NO | PRINTEX: | |

| | | | | | | | |
|------------|--------|--------------------|-----------------------------|---|--------|---------------|-------------------------------------|
| | BSR | PREFIX | OUTPUT PREFIX SPACES | V_ARG: | MOVE.L | {A2}+, - {A7} | MOVE BOTH ARGS AT ONCE |
| LDPH: | MOVE.L | #HEXDIGITS, AD | ADDRESS OF TRANSLATE TABLE | | BSR | CURSOR | POSITION THE CURSOR |
| | ROL.L | #4, D2 | PUT MSD IN LOW FOUR BITS | | ADDQ.L | #4, A7 | DROP BOTH ARGS |
| | MOVE.W | D2, D0 | PUT IN WORKING REGISTER | | BRA | LODP | CONTINUE COMMANDS |
| | AND.W | #\$000F, D0 | LEAVE ONLY LOW 4 BITS | * | | | |
| | MOVE.B | D{AD, DD.W}, DD | GET DIGIT FROM TABLE | * DEFAULT: | MOVE.W | D0, - {A7} | PRINT CHAR AS IS |
| | MOVE.W | D0, - {A7} | PUT ON STACK | | BSR | PUTC | |
| | BSR | PUTC | SEND IT | | ADDQ.L | #2, A7 | ADJUST STACK |
| | ADDQ.L | #2, A7 | DROP PARAMETER | | BRA | LOOP | DO NEXT |
| | DBF | D1, LDPH | LOOP UNTIL DONE | | PAGE | | |
| | BSR | POSTFIX | ADD TRAILING SPACES | * | | | |
| | BRA | LOOP | OO NEXT CONTROL CHAR | * CASE DISPATCH TABLE FOR PRINTF. * | | | |
| HEXDIGITS: | | | | DISPATCH: | DW | 9 | NUMBER OF VALID OPTIONS |
| | DB | '0123456789ABCDEF' | | | DB | D, 'd' | %d PRINT SIGNED DECIMAL W ADDRESS |
| S_ARG: | MOVE.L | {A2}, A0 | GET STRING ADDR FROM STACK | | LONG | O_ARG | |
| | CLR.W | D6 | GET STRLEN FOR FIELD ADJ | | DB | 0, 'u' | %u PRINT UNSIGNED DECIMAL W ADDRESS |
| SLEN: | TST.B | {AD}+ | LOOK FOR TERMINAL NULL | | LONG | U_ARG | |
| | BEQ | SK0_SA | BRANCH IF FOUND IT | | DB | 0, 'D' | %D PRINT SIGNED DECIMAL L ADDRESS |
| | ADDQ.W | #1, O6 | COUNT THIS CHAR | | LONG | D1_ARG | |
| | BRA | SLEN | LOOK AT NEXT CHAR | | DB | 0, 'U' | %U PRINT UNSIGNED DECIMAL L ADDRESS |
| SK0_SA: | MOVE.L | {A2}+, AD | GET STRING ADDR AGAIN | | LONG | U1_ARG | |
| | BSR | PREFIX | SEND LEADING SPACES | | DB | 0, 'x' | %x PRINT HEXADECIMAL W ADDRESS |
| LPD_SA: | TST.B | {AD} | END OF STRING? | | DB | 0, 'X' | %X PRINT HEXADECIMAL L ADDRESS |
| | BNE | SK1_SA | NO, KEEP PRINTING | | LONG | X1_ARG | |
| | BSR | POSTFIX | ALL DONE, SEND FINAL SPACES | | DB | 0, 's' | %s PRINT NULL-TERM STRING ADDRESS |
| | BRA | LOOP | JUMP OUT | | LONG | S_ARG | |
| SK1_SA: | MOVE.B | {AD}+, DD | GET CHAR FROM STRING | | DB | 0, 'c' | %c PRINT CHARACTER ADDRESS |
| | MOVE.W | D0, - {A7} | PUT ON STACK | | LONG | C_ARG | |
| | BSR | PUTC | AND SEND IT | | DB | 0, 'v' | %v SET CURSOR TO X,Y |
| | ADDQ.L | #2, A7 | ADJUST STACK | | LONG | V_ARG | ADDRESS |
| | BRA | LPD_SA | CONTINUE | | LONG | DEFAULT | UNKNOWN CASES HANDLED HERE |
| C_ARG: | MOVE.W | {A2}+, - {A7} | GET ARGUMENT | | | | |
| | BSR | PUTC | SEND LITERAL CHAR | END | | | |
| | ADDQ.L | #2, A7 | DROP ARGUMENT | | | | |
| | BRA | LOOP | NEXT COMMAND | | | | |

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you what you want!

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2
Copyrighted © by R. Jones & CPI

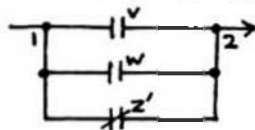
SOLUTIONS TO TEST 15

1.

$$\begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \begin{pmatrix} 1 & w & z' & v \\ 2 & w & 1 & x+y & x'y' \\ 3 & z' & x+y & 1 & w'+z \\ 4 & v & x'y' & w'+z & 1 \end{pmatrix} \end{array}$$

$$\begin{array}{c} 1 \quad 3 \quad 4 \\ \begin{pmatrix} 2 & w & x+y & x'y' \\ 3 & z' & 1 & w'+z \\ 4 & v & w'+z & 1 \end{pmatrix} \end{array}$$

$$\begin{aligned} f_{12} &= \underline{w} + (x+y)X(w'+z)v + x'y'(w'+z)z' + \underline{w(w'+z)} + (x+y)z' + x'y'v \\ &= w + vx'y' + v(x+y)X(w'+z) + z'(x+y) + x'y'z'(w'+z) \\ &= w + v(\underline{x'y'} + \underline{(x+y)X(w'+z)}) + z'(\underline{(x+y)} + \underline{x'y'(w'+z)}) \\ &= \underline{w} + vx'y' + \underline{vw'} + vz + xz' + yz' + \underline{wz'} \\ &= w + \underline{vx'y'} + \underline{v} + \underline{vz} + \underline{xz'} + \underline{yz'} + \underline{z'} = w + v + z' \end{aligned}$$

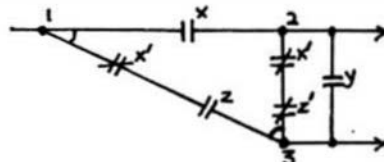


2.

$$\begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \quad 5 \\ \begin{pmatrix} 1 & x & 0 & 0 & z \\ 2 & x & 1 & y & z' & 0 \\ 3 & 0 & y & 1 & 0 & x' \\ 4 & 0 & z' & 0 & 1 & y' \\ 5 & z & 0 & x' & y' & 1 \end{pmatrix} \end{array}$$

$$\begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \\ \begin{pmatrix} 1 & x & x'z & yz \\ 2 & x & 1 & y & z' \\ 3 & x'z & y & 1 & x'y' \\ 4 & yz & z' & x'y' & 1 \end{pmatrix} \end{array}$$

$$\begin{array}{c} 1 \quad 2 \quad 3 \\ \begin{pmatrix} 1 & x & x'z \\ 2 & x & 1 & y + x'z' \\ 3 & x'z & y + x'z' & 1 \end{pmatrix} \end{array}$$



The final network is simpler if transfer-contacts are being used. The number of contacts remains at 6, but the number of springs is reduced from 12 to 10.

3.

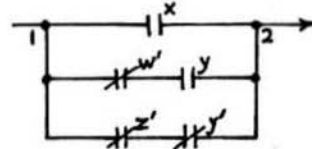
$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{pmatrix} 1 & x & 0 & z' & y & 0 \\ x & 1 & 0 & w' & 0 & y' \\ 0 & 0 & 1 & x' & z & z' \\ z' & w' & x' & 1 & 0 & 0 \\ y & 0 & z & 0 & 1 & w \\ 0 & y' & z' & 0 & w & 1 \end{pmatrix} & \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & x & 0 & z' & y \\ x & 1 & yz' & w' & wy' \\ 0 & yz' & 1 & x' & z+w \\ z' & w' & x' & 1 & 0 \\ y & wy' & z+w & 0 & 1 \end{pmatrix} \end{matrix}$$

$$\begin{matrix} & 1 & & 2 & & 3 & & 4 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 1 & & x & & wy+yz & & z' \\ & x & & 1 & & yz'+wy' & & w' \\ wy+yz & & yz'+wy' & & 1 & & x' \\ & z' & & w' & & x' & & 1 \end{pmatrix} \end{matrix}$$

$$\begin{matrix} & 1 & & 2 & & 3 \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} & 1 & & x+wz' & & wy+yz+xz' \\ & x+wz' & & 1 & & yz'+wy'+wx' \\ wy+yz+xz' & & yz'+wy'+wx' & & 1 & \end{pmatrix} \end{matrix}$$

$$\begin{aligned} f_{12} &= x + wz' + (wy + yz + xz'Xyz' + wy' + wx') \\ &= x + wz' + wyyz' + wywy' + wywx' + yzyz' + yzwy' + yzwz' + xz'yz' \\ &\quad + xz'wy' + xz'wz' \\ &= x + \underline{wz'} + \underline{wx'yz} + \underline{xy'z'} + \underline{wx'yz'} + \underline{wx'z'} \\ &= x + wz' + wx'yz + xy'z' \\ &= x + \underline{wz'} + \underline{wy} + \underline{yz'} \\ &= x + wz' + wy + yz' \end{aligned}$$

Verify this on a K-map



In problems of this sort there's nothing to prevent our using Pivotal Condensation to get us to a fourth-order matrix, and then switching to Laplace's Development to extract a third-order determinant, from which a final Boolean expression may be easier to produce.

Mile 21 - heading for Mile 22

Overnight I've been thinking long and hard about whether I should give you a word-logic problem, and decided that the going is tough enough right now that I shouldn't really add to your burden. What's that? You'd really like one? W-c-e-l-l ... if you INSIST, what else can I do? So here goes, but don't forget - you ASKED for it. Gather round, everyone, while your Uncle Bob unfolds a tale of mystery for you!!

A LOGIC PROBLEM IN THE LAND OF BINGONGO

Once upon a time there lived a man called Joe, and on his 21st birthday he was given a map by his father, which had been given to HIM by HIS father before him, and HIS father before him, and so on for several generations. This map showed where King Solomon's treasure was buried in the Lost City in the heart of the jungles of Bingongo. Joe's family had always been VERY poor, and could NEVER raise the funds to finance an expedition to recover this treasure, and it looked as though Joe, in turn, would have to hand the map on to HIS son - if he should ever have one.

Joe, however, figured that SOME treasure was better than none at all, so he invited some of his wealthier friends to finance the trip, in return for which he would share the treasure. And so this small group set off for Bingongo!

In time they came to the mouth of the mighty Bango-bango River, hired a large dug-out canoe and a team of paddlers, and began the several days' journey up-river to Bingongo itself. As the paddlers paddled rhythmically along, Joe studied the map and committed it to memory, as well as the instructions that came with it. He knew that he had to locate a mountain with twin peaks, at the foot of which his jungle path would fork into two, and then he had to take the LEFT fork. This was VERY important, for if he took the RIGHT fork he would eventually arrive at the village of a tribe of fierce cannibals, and would surely be eaten. This presented no problem to Joe, as he'd THOROUGHLY memorised everything!

He knew, too, that in the vicinity of the fork lived two different tribes of friendly natives, the Bingo tribe and the Bongo tribe, who were actually SO friendly that they had a custom of appointing an information-officer to sit at the fork in the road, and answer questions put to him by travellers. Each day an officer was selected at random from the population of either of the two tribes, so there'd be no way for a stranger to know whether the info-guy was a Bingo or a Bongo. Further, all questions HAD to be framed in such a way that he could answer with a simple YES or NO, otherwise no answer at all, and only ONE question per traveller (or group of travellers) was allowed. No problem there, it seems, except for one peculiarity! Bingos would ALWAYS answer truthfully, while Bongos would ALWAYS lie. Don't forget, only Bingos and Bongos could tell each other apart; they looked and behaved identically as far as a stranger was concerned!!

Joe wasn't worried by all this, however, as he KNEW which road to take - the one to the LEFT. Unfortunately, as they were paddling upstream, the canoe hit a crocodile and overturned, tipping everyone into the river, where they all got drowned or eaten by crocs, EXCEPT for Joe, who managed to swim to shore, losing his map in the process. There he came down with swamp-fever, and by the time he recovered from his delirium he only vaguely remembered the instructions, and wasn't at all sure which fork he had to take.

Still, as he'd come so far, he decided to press on to the Lost City, and eventually arrived at the fork in the road, and came face-to-face with the info-fella for the day. Here was a problem indeed!! How could he find out which fork to take when he couldn't be sure whether the answer he received was the truth or not? And remember, he was allowed only ONE question, which could only be answered with a YES or a NO.

| MAN | ROAD | REPLY | DESIRED |
|-----|------|-------|---------|
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 |

Joe was lucky! He worked it out with those little 1s and 0s. But how would YOU do it if you were Joe and found yourself in this predicament? What question would YOU put to the Bingo, or maybe he's a Bongo?? This isn't a TEST question, so just amuse yourselves with it, and I'll give the answer later!

Now let's get moving, as we've dillyed and dallied enough here. No more morning bed-time stories for a little while! Just for a change in menu, however, we'll leave Boolean matrices for a little while, and look instead at

RANDOM-INPUT SEQUENTIAL CIRCUITS

THE PRIMITIVE FLOW-TABLE

Unlike the sequential circuits discussed earlier, where the sequence follows a prescribed pattern laid down by the specs, random-input networks have no set pattern laid down. The specs merely detail what the output conditions will be IF the inputs happen to follow a certain sequence or arrive at a particular combination. Here's a typical example

"A circuit is to have two inputs, X1 and X2, and one output, Z. Z is to turn ON when X2 turns ON, provided X1 is ON at the time. Z will turn OFF when X2 turns OFF. Only one input will change state at a time."

Although this is a very simple case, the problem here is that the inputs occur in a random pattern, and we have to be certain that we cover all possible sequences for our machine to function correctly. We do this by drawing up a "primitive flow-table" with four columns headed for all possible combinations of X1 and X2, that is, 00, 01, 11 and 10, and an output column to the right headed Z. An opening row-1 is formed to get things going.

| x_1, x_2 | 00 | 01 | 11 | 10 | z |
|------------|----|----|----|----|-----|
| ① | 2 | - | 3 | 0 | |
| 1 | ② | 4 | - | 0 | |
| 1 | - | 5 | ③ | 0 | |
| - | 2 | ④ | 3 | 0 | |
| - | 6 | ⑤ | 3 | 1 | |
| 1 | ⑥ | 5 | - | 1 | |

Diagram 112

We'll begin in column-00, row-1 (which we refer to as address 00-1) with stable-state-1, circled to indicate a stable state, and a 0 under Z to keep track of the corresponding output. Address 11-1, which represents a double change of variables, is forbidden to this row, so a "-" is inserted in this position. Our procedure is similar to that used to create an iterative prototype network, in that we'll complete this row, creating new row-numbers as they become necessary, and then move down to complete these new rows in turn, until no new rows are called for.

The one guiding principle is that

"Each time we move horizontally from a stable state, and there CAN only be one stable state per row, we MUST move vertically in the new column to a stable state having the output condition called for. If such a stable state does not exist, then we MUST create a new row-number. Further, the output condition appearing in the Z-column belongs only to the stable state appropriate to that row."

When I say "stable-state" I am, of course, referring to a "circled" state. In row-1, for instance, the output 0 belongs to stable-state-1, and NOT to the unstable states appearing in the same row.

COMPLETING THE TABLE

OK then, let's commence our synthesis procedure by moving into column-01 (that is, we'll operate X2). The specs call for no output as a result of this change, and as we don't have a stable state with 0-output in this column, we'll create one by moving downwards to stable-state-2, with a 0 in column-Z. Similarly, if we consider a move to column-10 (X1 operated) instead, we're compelled to create a new stable-state-3 in this column, again with 0-output in column-Z. This completes row-1, but we now have two additional rows to deal with.

We'll tackle row-2 first, and enter a "-" in column-10, as this represents a double-change of variables from our stable-state-2. Next, if we move to column-00, the output must still be 0, and as we already have a stable state in this column with the required output condition, namely 1, we'll make the sequence return to that state by inserting a 1 at address 00-2. If we were to move to column-11 (we must be careful as this represents X1 FOLLOWING AFTER X2) the output must remain at 0, forcing us to create a new stable-state-4 for this column, with Z set to 0.

So we still have two rows to complete, and our first step in row-3 will be to place a "-" in column-01, after which we'll consider a move to column-00. There's no problem here, as we simply return to stable-state-1, but a move to column-11 (which represents X2 following after X1) calls for an output at last. Here, the only stable state in this column is 4, with 0-output, so we must create a new row-5 to accommodate this new condition.

And so we proceed in this fashion until the table naturally comes to a halt in row-6, at which point, PROVIDED ALL ADDRESSES HAVE AN ENTRY OF SOME KIND, we'll have been forced by the system of ONE STABLE STATE PER ROW to consider ALL possible sequences that the inputs might follow. Without this variant of our original design method we'd be unable to guarantee this state of affairs.

"POWER-ON" OUTPUT STATE

In the example just completed, we assumed, in the absence of any instructions to the contrary, that the "power-on" output state (that is, the state of the output at the moment of switching on our circuit) should be 0, and we therefore set Z to 0 in this row. Fortunately for us, a satisfactory flow-table resulted, but this won't always be the case, as the following example illustrates.

"A sequential circuit has two inputs, X1 and X2, and two outputs, Z1 and Z2. The input state $X1.X2 = 11$ is impossible (ie, X1 and X2 can NEVER be operated simultaneously), and only one input can change state at a time.

When $X1.X2 = 01$ or 10

then $Z1.Z2 = 00$

When $X1.X2 = 00$ following immediately after 01, then $Z1.Z2 = 01$

When $X1.X2 = 00$ " " " 10, then $Z1.Z2 = 10$ "

| $X1.X2$ | | | | |
|---------|----|----|--|---------|
| 00 | 01 | 10 | | $Z1.Z2$ |
| ① | 2 | 3 | | 00 |
| 4 | ② | - | | 00 |
| 5 | - | ③ | | 00 |
| ④ | 2 | 3 | | 01 |
| ⑤ | 2 | 3 | | 10 |

Diagram 113

If we choose the power-on stable-state-1 to have an output $Z_1Z_2 = 00$, the primitive flow-table of Diagram 113 results. Note that once the circuit action leaves row-1 it can NEVER return to that row unless power is switched OFF, so that it serves only as a power-on state, unlike our first example, where we made continuing use of this row during the sequencing of our circuit.

Unless they're specifically called for, such additional stable states should be avoided, as they generally lead to unnecessarily complicated circuits.

REDUNDANT STABLE STATES

Normally, if you stick strictly to the procedure set out, there won't be any redundant stable states, but in case you didn't completely follow the rules or if you come across a primitive flow-table designed by someone following a different method (and there are other methods) you'll need to know how to recognise and eliminate these undesired stable states from the flow-table.

The equivalence, or otherwise, of two rows in a primitive flow-table is based on the same set of rules that we applied to iterative prototype charts, and can be re-stated for random-input flow-tables as follows

"Provided the output states are compatible, that is, the rows belong to the same or an equivalent output group, two stable states are equivalent and can be combined into one row (taking the row-number of the lower of the stable-state-numbers) if they meet the following conditions

- (a) They occur in the same column, and
- (b) The UNSTABLE states in both rows are identical, or are themselves equivalent or pseudo-equivalent."

FORMING A "COMBINED" FLOW-TABLE

| X_1X_2 | | | | | X_1X_2 | | | | |
|----------|----|----|----|----------|----------|----|----|----|----------|
| 00 | 01 | 11 | 10 | Z_1Z_2 | 00 | 01 | 11 | 10 | Z_1Z_2 |
| ① | 2 | - | 3 | 00 | ① | 2 | - | 3 | 00 |
| 1 | ② | 4 | - | 10 | 1 | ② | 4 | - | 10 |
| 5 | - | 4 | ③ | 11 | 1 | - | 4 | ③ | 11 |
| - | 6 | ④ | 3 | 01 (a) | - | 6 | ④ | 3 | 01 |
| ⑤ | 7 | - | 3 | 00 | 1 | ⑥ | 4 | - | 11 |
| 5 | ⑥ | 4 | - | 11 | | | | | |
| 1 | ⑦ | 4 | - | 10 | | | | | |

Diagram 114

In the primitive flow-table of Diagram 114a, stable-states 2 and 7 are equivalent, as they occur in the same column-01, the output states are in the same group-10, and the unstable states are identical. Stable-states 1 and 5, on the other hand, depend for their equivalence on the equivalence of stable-states 2 and 7, and as we've already established the equivalence of this pair, it follows that 1 and 5 are also equivalent.

We may therefore combine rows 1 and 5 under the new row-number 1, and rows 2 and 7 under the new number 2, to produce the "combined" flow-table of Diagram 114b.

| X_1X_2 | | | | | X_1X_2 | | | | |
|----------|----|----|----|----------|----------|----|----|----|----------|
| 00 | 01 | 11 | 10 | Z_1Z_2 | 00 | 01 | 11 | 10 | Z_1Z_2 |
| ① | 2 | - | 3 | 00 | ① | 2 | - | 3 | 00 |
| 5 | ② | 4 | - | 10 | 5 | ② | 4 | - | 10 |
| 5 | - | 4 | ③ | 11 | 5 | - | 4 | ③ | 11 |
| - | 6 | ④ | 3 | 01 | - | 6 | ④ | 3 | 01 |
| ⑤ | 7 | - | 3 | 00 | ⑤ | 7 | - | 3 | 00 |
| 5 | ⑥ | 4 | - | 11 | 5 | ⑥ | 4 | - | 11 |
| 1 | ⑦ | 4 | - | 10 | 1 | ⑦ | 4 | - | 10 |

Diagram 115

In the primitive flow-table of Diagram 115, the equivalence of stable states 1 and 5 depends on the equivalence of stable states 2 and 7. In turn, the equivalence of states 2 and 7 depends on the equivalence of states 1 and 5. Here we've come full circle to our starting point without meeting up with any non-equivalence, so each of these pairs of stable states can be combined to reduce the primitive flow-table to five rows.

You should be VERY careful to distinguish between COMBINING rows, which still leaves only ONE stable state per row, from MERGING of rows in our normal flow-table, where it's possible to have more than one stable state in a row.

The interdependence of equivalences just discussed can be continued over many rows, provided that the chain does not end in a definite non-equivalence. Put in another way, it can be said that

"Two stable states occurring in the same column and having the same, or equivalent, output conditions can be made equivalent, unless the equivalence depends on a non-equivalence."

Sounds a little complicated, but if you read it slowly it should be fairly clear!

A SYSTEMATIC METHOD FOR COMBINING ROWS

x_1, x_2

| | 00 | 01 | 11 | 10 | z_1, z_2 |
|---|----|----|----|----|------------|
| ① | 2 | 4 | 3 | | 00 |
| 6 | ② | 5 | 3 | | 11 |
| 1 | 7 | 4 | ③ | | 01 |
| 9 | 2 | ④ | 8 | | 10 |
| 1 | 10 | ⑤ | 3 | | 10 |
| ⑥ | 7 | 5 | 11 | | 00 |
| 1 | ⑦ | 12 | 3 | | 11 |
| 1 | 7 | 4 | ⑧ | | 11 |
| ④ | 2 | 12 | 11 | | 00 |
| 9 | ⑩ | 4 | 11 | | 11 |
| 9 | 10 | 12 | ⑪ | | 01 |
| 1 | 2 | ⑫ | 8 | | 10 |

Diagram 116

Diagram 116

In a complicated flow-table, the various equivalences and non-equivalences are often extremely difficult to establish, and a systematic approach is the only solution. Consider the primitive flow-table of Diagram 116. In column-00, 1, 6 and 9 are possible equivalences because they belong to the same output group. In column-01, 2, 7 and 10 are possible. In column-11, 4, 5 and 12 are possible equivalences, while in column-10 only 3 and 11 are possible, 3 and 8, and 8 and 11 being immediately established as non-equivalences. Bear in mind that all these are just POSSIBILITIES, as it still has to be determined whether they are in fact equivalences.

The systematic approach to this problem can be applied with equal success to the establishing of equivalences in the rows of an iterative prototype table, and should definitely be used whenever a complicated table appears.

| | 1-6 | 1-9 | 6-9 | 2-7 | 2-10 | 7-10 | 4-5 | 4-12 | 5-12 | 3-11 |
|--------|-----|-----|-----|-----|------|------|-----|------|------|------|
| X 1-6 | | | | ✓ | | | | | | |
| 1-9 | | | | | | ✓ | ✓ | ✓ | | ✓ |
| X 6-9 | | | | | ✓ | | | | | |
| X 2-7 | ✓ | | ✓ | | | | | | | |
| X 2-10 | | | | | | | ✓ | | ✓ | |
| 7-10 | | | | | | | | | | ✓ |
| X 4-5 | ✓ | | | | ✓ | | | | | |
| 4-12 | | ✓ | | | | ✓ | | | | ✓ |
| X 5-12 | | | ✓ | ✓ | | | | | | |
| 3-11 | ✓ | ✓ | | | ✓ | ✓ | | | | |
| X 3-8 | | | | | | | ✓ | | ✓ | |

Diagram 117

A table is first drawn up, as in Diagram 117, with the columns headed with all possible equivalence-PAIRS, and a set of rows with identical pair labels. Now let's start in column 1-6, and consider the equivalence of stable states 1 and 6. Looking at rows 1 and 6 in the primitive flow-table, we see that they depend for their equivalence on the equivalence of rows 2 and 7, 4 and 5, and 3 and 11, and so, in column 1-6 of the equivalence-table, we place a tick in the ROWS which bear these number-pairs, that is, 2-7, 4-5 and 3-11. And so on for each of the other columns.

I'd like to mention column 4-5, where the equivalence of these two states depends not only on the possible equivalence of 1-9 and 2-10, but also on the known non-equivalence of 3-8. This necessitates adding another row for 3-8 below the rest of the table, separated from it by a solid line, and with a tick in column 4-5. When we come to completing column 5-12, we find that we have to make use of this row once more.

Now for the easy part! Because columns 4-5 and 5-12 depend for their equivalence on a known non-equivalence, THEY TOO MUST BE NON-EQUIVALENT. We record this fact by entering one diagonal of a small "x" to the left of ROWS 4-5 and 5-12. Scanning along row 4-5, we find that 1-6 and 2-10 depend on this new non-equivalence, and therefore cannot be equivalent either. This fact is recorded by entering one diagonal of an "x" to the left of rows 1-6 and 2-10, and at the same time completing the "x" of row 4-5 to signify that we've checked out this row.

Having disposed of row 4-5, let's do the same thing with row 5-12. Studying ROW 5-12, we are forced to place half-an-x in rows 6-9 and 2-7, as it's these columns which depend on row 5-12 for THEIR equivalence, and also complete the "x" in row 5-12, which means "this row completed".

Now, picking half-x rows in sequence, we see that in row 1-6 only 2-7 depends on it, and as this row already has a half-x, we simply make 1-6 into a complete-x. Similar remarks apply to row 6-9 and the half-x already in row 2-10, so 6-9 ends up with a complete-x. Row 2-7 has the two columns 1-6 and 6-9 dependent on it, but these two rows have already been checked out with a complete-x, so we can complete this row's "x". Finally a similar situation exists with row 4-5.

When all is done we have the exact table of Diagram 117, with rows 1-9, 7-10, 4-12 and 3-11 the only rows NOT marked with an "x". These four pairs have thus been established as definite equivalences, and may therefore be combined into four singles, and the primitive flow-table reduced to eight rows!

OPTIONAL OUTPUT STATES

Sometimes maximum stable state reduction demands that a particular stable-state be made equivalent to two or more other stable-states, which are themselves non-equivalent. This very often applies when optional OUTPUT-states are specified, as in the primitive flow-table of Diagram 118a. Here stable-states 4 and 6 are non-equivalent (they belong to different output groups), thereby making stable-states 5 and 7 non-equivalent.

| X_1X_2 | | | | Z |
|----------|----|----|---|-----|
| 00 | 01 | 10 | | |
| ① 2 | 3 | | 0 | (a) |
| 1 ② 3 | | | 0 | |
| 1 2 ③ | | | 1 | |
| 1 ④ 5 | | | 0 | |
| 1 6 ⑤ | | | 1 | |
| 1 ⑥ 7 | | | 1 | |
| 1 4 ⑦ | | | 1 | |

| X_1X_2 | | | | Z |
|-------------------------------|----|----|---|-----|
| 00 | 01 | 10 | | |
| ① $\frac{2}{6}$ $\frac{3}{7}$ | | | 0 | (b) |
| 1 ④ 5 | | | 0 | |
| 1 6 ⑤ | | | 1 | |
| 1 ⑥ 7 | | | 1 | |
| 1 4 ⑦ | | | 1 | |

Diagram 118

Let's check out stable-state-2, with a phi output, against stable-state-4. We see that they can be made equivalent if 3 and 5 are equivalent, which in turn depends on the equivalence of 2 and 6, which depends on 3 and 7, and finally on 2 and 4. So we've come full-circle, without meeting up with a single non-equivalence along the way. Notice that in this loop we've checked out not only the original 2-4 pair, but also the 2-6 pair, so stable-state-2 is equivalent to both 4 and 6, though, as we observed earlier, 4 and 6 are not themselves equivalent! In the same way, stable-state-3 is equivalent to both 5 and 7, although THEY are not equivalent.

The combined flow-table for these conditions is shown in Diagram 118b, from which we see that the UNstable states 2 and 3 in the first row can each be replaced by either of two entries, though at this stage it's not possible to say which choice will lead to the "best" circuit, and we'd therefore have to design four different ones and then take our pick.

An excellent guide, though I've not done enough tables to be sure that it'll ALWAYS work (it hasn't let me down yet!) is to look for another row in the same output group, and make the numbers agree with this row. In this example, row-2, stable-state-4, is entirely compatible, and my choice would therefore be to make row-1 read 1 4 5. We'll complete this study next time, but in the meantime I'll give you just two problems to see whether you've got all this tucked away in your personal RAM. You've also got the Bingongo problem to think about till then. Here then is

TEST SIXTEEN-A

1. A sequential circuit has two inputs, X1 and X2, representing, in binary, the decimal numbers 0 to 3 inclusive, and one output, Z. If a change in input INCREASES the represented number by one, the output is to turn ON, if not already ON. If an input change DECREASES the represented number by one, the output is to change state. Any other input change will leave the output unchanged. All input changes are possible. Take careful note that the specs refer to an input change of one unit only, that is, from 1-2 or 2-3 or 1-0, etc. Develop a primitive flow-table, and combine rows if possible.

2. Refer to the problem of Diagram 113 in our discussion of "power-on" output states, and then develop the primitive flow-tables if the initial "power-on" output state is (i) 01 (ii) 11 and (iii) 10.

On the next leg of our journey, we'll discover how to "merge" (as distinct from "combine") the combined flow-table, and then how to convert this table to our standard form, so we can complete the design process and produce a finished circuit-diagram. See you then!

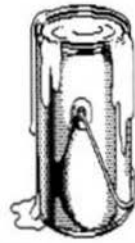
... End of Mile 21. Now at marker Mile-22.

+++

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Bit-Bucket



By: All of us

"Contribute Nothing · Expect Nothing", DMW '86

Microware Systems Corporation

David F. Davis
515/224-1929

Davis-Marrin Communications
Ken Marrin
619/721-5259

OS-9 REAL-TIME OPERATING SYSTEM AVAILABLE FOR 68030, 68070 AND MVME 147 CPU BOARD

DES MOINES, Iowa. Buscon/89-West—Microware Systems Corporation announces the availability of the OS-9 Operating System and development environment for Signetics' 68070 microcontroller, Motorola 68030 microprocessor and MVME 147 single-board computer.

Microware will provide a specially tailored version of the OS-9 Kernel to support the 84-pin SCC68070 microcontroller, which consists of a 68000 compatible processor, memory management unit, UART, two-channel DMA controller, two timers and a clock. This version of OS-9 will include an assembler, high-level C compiler, 'zIHu'zIOMACS screen editor and both ROM and symbolic assembly language debuggers, plus drivers and file managers to support all of the 68070's on-chip peripherals. By combining the appropriate OS-9 system modules with user-developed applications modules, designers can configure ROM-based 68070/OS-9 systems using less than 32Kbytes of ROM and 128Kbytes of RAM.

Microware is also providing a customized version of OS-9 to support the 68030. This kernel supports the 68030's on-chip memory management and cache, as well as an expanded external cache. It also includes a MATH trap module that supports both hardware (MC68881 or MC68882) and software floating point arithmetic.

To support the 68030's memory management unit, OS-9 also includes a System Security Module (SSM). The SSM uses a system-wide permission scheme to limit RAM accesses for user-state

processes. When a process attempts to access memory for which it does not have access permission, OS-9 generates a recoverable bus error.

The first 68030-based CPU board that Microware is supporting is Motorola's 68030-based MVME 147 single-board computer. OS-9 for the MVME 147 includes the OS-9 Kernel, assembler, high-level language compilers (C and BASIC), 'zIHu'zIOMACS screen editor and both ROM and symbolic assembly language debuggers. The Motorola MVME 147 also features on-board Ethernet, SCSI and 68881/68882 floating point. Through its file managers, this optimized version of the OS-9 Kernel supports all of the MVME 147's peripheral devices.

To make OS-9 available to its MVME 147 customers, Schweber Electronics will distribute OS-9 with the MVME 147 as a bundled solution. This will mark the first availability of a bundled real-time OS for Motorola hardware.

Both the MVME 147 and Signetics' Microcore boards can be used as development platforms, or integrated with Microware's popular UniBridge and PCBridge cross-development packages. These packages provide a comprehensive development environment for distributed programming/supervisory control.

CPU BOARDS WITH INTEGRATED LAN HARDWARE ENABLES 680X0-BASED HOST CPU BOARD TO RUN TCP/IP WITHOUT AN EXTERNAL COMMUNICATIONS CONTROLLER BOARD

Des Moines, Iowa. Uniform, 1989—Microware announces a network communications software package that enables a host CPU board with integrated LAN hardware (Such as Motorola's MVME147, which includes the LANCE Ethernet chip set) to run the TCP/IP protocol without requiring an external communications controller board. Known as the OS-9/ Internet Support Package (OS-9 /ISP), this ROMable software enables any OS-9 based 680X0 host to communicate with any other system (such as

Unix, VMS, or DOS based systems) running the TCP/IP protocol.

Initially, OS-9/ISP will provide drivers for the LANCE Ethernet chip set. However, the software's device independence makes it adaptable to a variety of hardware configurations, from ArcNet and SLDC, to point-to-point serial links.

OS-9/ISP conforms fully with the DARPA Transmission Control Protocol/Internet Protocol (TCP/IP). It also supports the DARPA specified File Transfer Protocol (FTP) and TELNET (virtual network terminal, also known as remote login). These protocols not only enable OS-9 users to access other nodes on the network, but enable other nodes to access an OS-9/ISP system as a server.

An OS-9 BSD 4.3 Unix socket-style communications interface known as "sockman" provides the link between application programs and the TCP/IP protocol. Sockman is a superset of the enpman socket manager that Microware provides with its Ethernet Support Package (A TCP/IP package that runs on dedicated communications controller boards such as the one available from CMC).

As a result, applications written for dedicated communications controllers under enpman will run without modification on a host CPU board under sockman. Address family protocols (Internet, for example) are implemented by sockman as subroutine modules. This modular design enables designers to easily add and dynamically load new address families such as X.25.

The Interface Manager, or "ifman", provides the socket protocol modules for the hardware-independent layer that handles system configuration and diagnostics. This package enables the system configuration (such as changing a station's address, or adding/deleting stations) to be modified dynamically without requiring driver recompilation. Because it is hardware independent, this layer enables a system to support multiple network interfaces, such as Ethernet, ArcNet, SLDC, and point-to-point serial links.

OS-9/ISP is available pre-installed and pre-configured for Motorola's MVME147. It is distributed in object code form on diskette and includes file managers (sockman, ifman, and pkman), the Internet protocol handler modules, network device drivers and descriptors, configuration modules, utility programs, and socket programming interface libraries.

To simplify the porting effort for system integrators who wish to port OS-9/ISP to their own boards, Microware provides an OS-9/ISP Portpak. Distributed on either diskette or tape, Portpak provides the OS-9/ISP object code modules, as well as the source code for a sample driver and documentation that simplifies the adaptation and development of drivers for a particular link-level network interface.

The cost for OS-9/ISP running on a single node OS-9/680X0-based system is \$650.00. Quantity discounts are available.

OS-9 SUPPORTS BACKPLANE-BASED MULTIPROCESSING OVER VME BUS

DES MOINES, Iowa. Microware Systems Corporation announces an extension to its Network File System that supports multiprocessing via RS232C cable, local area networks such as Arcnet (Ethernet drivers will be available soon) and the VMEbus system backplane. Known as OS-9/NET, this software provides a logical extension to the OS-9 I/O system, enabling users to access files and devices resident on remote systems as if they are resident on the user's own system.

OS-9/NET works with any peripheral device, including other CPU's, printers, and mass storage devices. It supports all of the functions offered in the standard OS-9 disk file manager, such as creating and deleting files and directories, and changing working directories.

Files and devices are accessed or "opened" from a remote system by adding a logical system name to a standard OS-9 pathlist. For example, to list the file "read.me" resident on the remote system mware1, the user would write:

```
list /net/mware/h0/readme.
```

The network file managers on the local and remote nodes automatically and transparently convert all input/output requests to the necessary internal logical and physical network protocols. OS-9/NET supports five network utility programs:

- | | |
|--------|---|
| chp | Change Processor command. CHP starts a remote shell on a specified node. |
| mtsmon | Multiple Time-Sharing Monitor. Mtsmon supervises idle terminals and initiates the login sequence for time sharing applications. |
| ndir | Network Directory command. Ndir checks and displays the status of network stations. |
| nmon | Network Monitor. Nmon allows stations to enter or leave the network and handles network permissions. |
| nwatch | Network Watchdog. Nwatch monitors the reliability of network stations. |

An additional level of file security positively controls network access to files on each node.

OS-9/NET is available immediately.

For more information, please contact Microware Systems Corp., 1900 N.W. 114th St., Des Moines, Iowa 50322. Phone (515) 224-1929. Microware offices are located in Des Moines, Iowa; Santa Clara, California; Southampton, UK; and Tokyo, Japan, with field representatives worldwide.



MOTOROLA INC.

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-8598

CONTACTS
Sherena Dover
Cunningham Communication, Inc.
(408) 982-0400

Don Mackey
Motorola Inc.
(512) 891-2839

MOTOROLA UNVEILS 68040 ARCHITECTURE

040 Raises Stakes in Microprocessor Battle
Breaks One Million Transistor Barrier
Hewlett-Packard First to Endorse Chip

AUSTIN, Texas, March 28, 1989 — Motorola's Microprocessor Products Group today raised the stakes in the microprocessor market by unveiling architectural features of its 32-bit 68040 (040). The new chip, the latest member of Motorola's 68000 microprocessor family, is the first to incorporate more than 1.2 million transistors on a single piece of silicon, making it the most integrated microprocessor in the industry.

This integration, coupled with the chip's multiple execution units, will establish the 040 as the fastest complex-instruction set microprocessor ever, outperforming all 32-bit competing architectures and many RISC products. It is also the first conventional microprocessor to include floating point on board. Additional 040 features, performance, pricing and availability will be made public later this year.

The 68040 represents a dramatic leap in 68000 technology, far greater than the increase from the 68020 to the 68030. Its architecture incorporates more than five major units on chip, all of which operate simultaneously to deliver very high performance. The main features of the 040 include an integer unit, a floating-point unit, a memory management unit and separate caches for data and instructions (see diagram).

In a related announcement, Hewlett-Packard said that it will incorporate the 040 in top-of-the-line workstations, making it the first company to endorse the new chip. Hewlett-Packard, the world's second largest manufacturer of workstations, also said that it will provide 040 development tools.

The 040 maintains 100 percent compatibility with Motorola's 68000 family, inheriting a \$3 billion software base and providing an upgrade path for a \$100 billion hardware base. This compatibility allows more than 1,000 companies to incorporate the new chip. According to InfoCorp, a market research firm based in Santa Clara, Calif., the 68000 architecture powers more than 63 percent of all computer systems priced from \$12,000 to \$300,000.

"The 040 is an architecture that our competitors will be hard pressed to match," said Jack W. Browne, director of marketing of Motorola's Microprocessor Products Group (Austin, Texas). "Coupled with our vast lead in 32-bit software, the 040 expands the 68000 family's dominance in advanced personal computer, workstation and multiuser markets."

The 040's architecture is maximized for a high degree of parallel operation through the use of the five execution units, multiple pipelines and internal buses as well as a full Harvard-style architecture. Each of the five units on the chip is described below.

Integer Unit: A major redesign with 100 percent software compatibility

The integer unit of a processor executes the instructions in a program. The 040's integer unit implements the 68030 instruction set, allowing it to access the world's largest 32-bit software base. While maintaining 100 percent compatibility with the 68000 family, the 040 integer unit includes features that significantly increase performance by reducing the time needed to execute instructions.

Floating-Point Unit: 80 bits of processing power

Floating-point units accelerate mathematical computations for applications such as graphics and financial analysis. Today, most floating-point processing requires separate chips.

The 040 includes this capability on chip by coupling its integer unit with a powerful 80-bit floating-point processor. The floating-point unit incorporates dedicated hardware that directly supports floating-point instructions to increase performance. The floating-point unit also conforms to IEEE Standard 754 and maintains compatibility with Motorola's 68882 (882) floating-point math coprocessor, often utilized with members of the 68000 family.

Memory Management Unit: Extending Motorola's lead in Unix®

Memory management is a crucial feature for running large operating systems such as Unix. Motorola's 68000 family dominates the Unix market.

The 040's on-chip paged memory management unit supports both demand-paged virtual memory and real-time operating systems. The MMU contains two independent address translation caches (ATC) that simultaneously translate both data and instruction addresses.

Caches: Feeding the machine

Caches provide very high-speed temporary storage of information recently used by the processor. They are especially important with high-performance microprocessors where execution units demand a constant flow of information. Motorola's earlier 68020 and 68030 microprocessors pioneered the use of on-chip caches to reduce delays and costs associated with large off-chip memory.

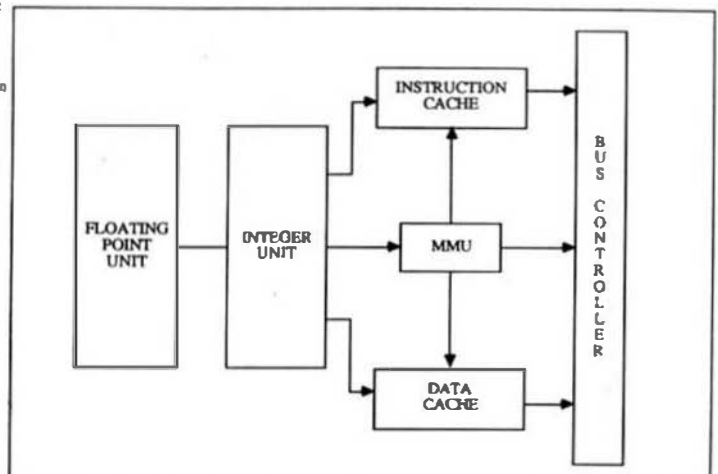
The 040 integrates large separate caches for instructions and data. These caches are the most efficient offerings in the industry and supply the execution units with a constant flow of information.

Multiprocessor Innovation: Delivering supercomputer performance

Multiprocessing allows systems to be configured around a series of microprocessors linked together to provide high system performance. The 040 also offers several new features that support multiprocessing designs. It includes a hardware mechanism called the wrap controller that controls the use of data in a multiprocessor array, ensuring that each processor uses the latest, correct version of data. Scoping maintains "cache coherency," thus preventing potential discrepancies that can result when several processors share and manipulate the same data.

Motorola's \$2.9 billion Semiconductor Products Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a part of Motorola Inc. The company is the largest and broadest supplier of semiconductors in North America, with a balanced portfolio of more than 50,000 devices.

PRELIMINARY 68040 BLOCK DIAGRAM



Integer Unit — The 040's integer unit implements the 68030's instruction set, giving it access to the world's largest 32-bit software base. The integer unit includes features that significantly increase performance by reducing the time needed to execute instructions. The 040 includes an internal bus structure that allows the integer unit to access instructions and data simultaneously.

Floating-Point Unit — The 040's on-chip floating-point unit offers 80 bits of processing power. The unit incorporates dedicated hardware that directly supports floating-point operations. The 040's floating-point unit conforms to IEEE Standard 754 and is compatible with Motorola's 68882 (882) floating-point math coprocessor commonly used with members of the 68000 family.

Memory Management Unit (MMU) — The paged-memory management unit supports both demand-paged virtual memory and real-time operating systems. The MMU is tightly coupled with the 040's on-chip instruction and data caches.

Caches — The 040 incorporates large separate caches for instructions and data. The caches are designed to supply the chip's execution units with a constant flow of information.

Like the 030, the 040 offers a full Harvard-style architecture. This architecture coupled with the chip's execution units, multiple pipelines and internal buses enables the 040 to deliver a high degree of performance.

Shemaz Daver
Cunningham Communications, Inc.
(408) 982-0400

Dick Muldon
AT&T
(201) 221-2694

Dean Mosley
Motorola
(512) 891-2839

Michael Carey
88open Consortium Ltd.
(503) 682-5703

MOTOROLA'S 88000 BCS ENDORSED AS DEVELOPMENT PLATFORM FOR UNIX SYSTEM V RELEASE 4/ABI

Current 88000 Software To Run on Future Unix System Versions

UniForum, San Francisco, Feb. 28, 1989 — Motorola and AT&T today announced that the 88000 Binary Compatibility Standard (BCS) will be used as an early development platform for AT&T's Unix® System V.4 Application Binary Interface (ABI). This ensures that current software for the 88000 will run on future versions of Unix System V without recompilation.

In April 1988, Motorola and AT&T signed an agreement to develop an ABI for the 88000 RISC architecture. AT&T's ABI only applies to the environment for Unix System V.4. To establish standards for Unix System V.3.2, Motorola, AT&T and the 88open Consortium Ltd. worked closely together to create the 88000 BCS that allows for software compatibility across 88000-based systems. This BCS is the first and only documented set of calling standards for Unix System V.3.2 and is a stepping stone to the ABI for Unix System V.4. The 88000 BCS incorporates select features of the ABI currently under development allowing for migration of software from System V.3.2 to System V.4. AT&T and Motorola are developing an ABI compliant version of Unix System V.4 for the 88000 which will be available to the industry in late 1989.

Last week, more than 30 companies announced the development of software based on the 88000 BCS including Frame Technology, Informix and WordPerfect. Motorola will begin shipment of development platforms based on the 88000 BCS in April.

"Our efforts with Motorola to develop an ABI for System V.4 version for the 88000 enhances the strength of Unix System V in the marketplace," said Mike DeFazio, director of Unix System Software at AT&T (Morristown, N.J.). "Our work with Motorola has brought us closer to establishing a standard operating system environment for open systems."

"Since the introduction of the 88000, we have adopted the strategy of developing software standards for our RISC architecture," said Murray Goldman, general manager and senior vice president of Motorola's Microprocessor Products Group (Austin, Texas). "Today's announcement with AT&T clearly validates our software strategy and cements our leadership position in the RISC market."

Motorola's 88000 RISC architecture is the most complete solution on the market with a floating point and integer unit on one chip (88100) and cache and memory management on a second (88200). Motorola has the most hardware products based on a RISC processor on the market today including systems from Data General, Everex and Seaborn. In the next few months, additional 88000-related announcements will be made.

READER CONTACT:
MOS Memory Marketing
512/928-6700

INQUIRY RESPONSE:
Technical Info Center
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA FAST STATIC RAMS OFFER MORE OPTIONS

Austin, Texas, February 22, 1989..... The Motorola Memory Products Division is adding two new CMOS fast static RAMs (SRAMs) to their extensive memory offering. To meet the evergrowing demand of system designers for more options, Motorola is now offering 4K x 4 and 16K x 4 fast SRAMs with a very fast output enable function.

The MCM6290P25/30/35, previously announced in SOJ (March 17, 1988) is now available in a 22 lead, 300 mil plastic dual-in-line package (PDIP). The MCM6290 is configured 16K x 4 and features fast output enable time of 12 ns for MCM6290P25 and 15 ns for MCM6290P30/35. The fast output enable function is ideal for 32-bit microprocessor systems that depend on separate cache memory to enhance overall memory performance.

In addition to the 16K x 4 in PDIP, Motorola is also introducing a 4K x 4 fast SRAM with very fast output enable. This device is also ideal for 32-bit microprocessor systems with separate cache memory and is especially cost effective in those applications where 4K of memory depth will suffice. The 4K x 4 device comes in a 24 lead, 300 mil small outline J-lead (SOJ) package and also is available in a 22 lead, 300 mil PDIP. The part numbers are MCM6270J25/35 for SOJ and MCM6270P25/35 for PDIP. Output enable access time is 12 ns for the 25 ns part and 14 ns for the 35 ns part.

Both the MCM6270 and MCM6290 are designed and manufactured with Motorola's high performance silicon-gate CMOS technology. The die for these devices are fabricated in Austin, Texas at the same facility where Motorola also fabricates their 32-bit MC68020 and MC68030 microprocessors.

The full line-up of Motorola x4 fast SRAMs with very fast output enable is as follows:

| PART NUMBER | ORG | ADDRESS/ CHIP ENABLE | | OUTPUT ENABLE | | 100 PIECE PRICE |
|-------------|---------|-------------------------|-------|------------------|-------|-----------------------|
| | | SPEED | SPEED | ENABLE | AVAIL | |
| MCM6270J25 | 4K x 4 | 25 NS | 12 NS | NOW | NOW | \$6.90 |
| MCM6270J35 | 4K x 4 | 35 NS | 14 NS | NOW | NOW | \$5.18 |
| MCM6270P25 | 4K x 4 | 25 NS | 12 NS | NOW | NOW | \$6.38 |
| MCM6270P35 | 4K x 4 | 35 NS | 14 NS | NOW | NOW | \$4.58 |
| MCM6290J25 | 16K x 4 | 25 NS | 12 NS | NOW | NOW | \$17.33 |
| MCM6290J30 | 16K x 4 | 30 NS | 15 NS | NOW | NOW | \$15.18 |
| MCM6290J35 | 16K x 4 | 35 NS | 15 NS | NOW | NOW | \$13.28 |
| MCM6290P25 | 16K x 4 | 25 NS | 12 NS | NOW | NOW | \$15.75 |
| MCM6290P30 | 16K x 4 | 30 NS | 15 NS | NOW | NOW | \$13.80 |
| MCM6290P35 | 16K x 4 | 35 NS | 15 NS | NOW | NOW | \$12.08 |

Motorola is a winner of the first annual Malcolm Baldrige National Quality Award, in recognition of its superior company-wide management of quality processes. Congress established the award to motivate U.S. companies to improve their worldwide competitiveness through quality strategies.

Computer Systems Consultants, Inc.
E. M. Pass, Ph.D., President
1454 Latta Lane Conyers, GA 30207
Telephone Number 404-483-4570/1717

Computer Systems Consultants was founded in 1972 and was incorporated in 1977. The company develops and markets personal and small-business software and hardware and provides consulting services.

Its primary means of advertising is through magazine coverage in several specialty magazines, including '68' Micro Journal, Color Micro Journal, Hot COCO, etc. Most of its customers are in the United States, but a considerable percentage may be found outside the United States. It has done business with over 1000 clients, many on a multiple-order basis. It has a network of over 50 dealers and distributors, on a worldwide basis.

Most of the hardware products and software products were developed for the Motorola 68xx line of microcomputer systems. Some of the products have subsequently been placed on the IBM Personal Computer and similar computers which operate under MSDOS and PCDOS, in addition to systems operating under UNIX and XENIX.



Intermetrics

Software Products Division

A division of the Development Systems Group of Intermetrics, Inc.

Address: 733 Concord Avenue
Cambridge, MA 02138
Telephone: (617) 861-1840
Press: Patricia Arcand
Contact: (617) 861-1840 ext. 4015

Business: Designs, markets and provides technical support for InterTools, integrated software development tools for the designers of microprocessor-based systems; and for the Whitesmiths' line of software development tools and the IDRIS operating system.

Product Line: • InterTools C cross-compilers, cross assemblers, XDB - source level cross debuggers, and utility programs for programming embedded microprocessors.
• Whitesmiths' C native and cross compilers, simulators, assemblers, and IDRIS, the first real-time operating system which was not derived from AT&T code.

Host Platforms: VAX, MicroVAX, VAXstation (VMS, Ultrix), Sun, Apollo, HP 9000 Series 300, IBM 370, IBM PC XT/AT and compatibles.

Targets: 68000/166/286, 68000/010/020, 68000/06/00, 68HC11, Z80, V20-V30, V80, and the Am29000 (Advanced Micro Devices).

Compatible Instruments: Emulators from Applied Microsystems, ZAX, Tektronix, Microtek International, Sophia, and Hewlett Packard.

Directors: William Carlson, General Manager
Dlem Le, Director of Sales and Marketing
Alan Johnson, Sales Manager
Jim Watkins, Product Marketing Manager



MICRO-LINK introduces STD207 Single Board System

The fastest 16 bit industrial microcomputer systems in the world are now being designed around the Micro-link STD207 Single Board System, the latest addition to an extensive line of STDbus products.

Using a 68000/010 CPU running at 16MHz, with 512K DRAM and up to 512K EPROM on board without wait states, the performance exceeds 2000 Dhrystones. The on board system facilities include two RS232/485 serial ports, a parallel printer port, a SCSI interface, three 16 bit counter/timers, and extensive interrupt capabilities.

Included with the card is the MICRObug debugger or, optionally, the STDPRON embedded real-time, multitasking PDOS operating system. Development and target system configurations for all industrial applications, together with software tools and support, are available from Micro-Link.

The STD207 SBS is a standard size STD card and is fully compatible with existing STD products, including over 100 Micro-Link boards and accessories. STD207 Products are available now, with card prices ranging from \$400 to \$1500 in quantity 100.

For further information and product guide, contact Jim Bull at 1-800-428-6155 or 317-846-1721, 14602 North US Hwy 31, Carmel, IN 46032.

SOFTWARE DEVELOPMENT SYSTEMS, INC.

Questions? Call 1-800-448-7733 or 1-312-971-8170
or send a FAX to 1-312-971-8513

—The Compiler—

Twelve major features
make your job easier

1. It's 100% ROMable

CrossCode C provides all the startup source code you need to set up your stack and initialize RAM. The startup code takes you all the way from "reset" to function "main", and you can customize it to suit your application.

The compiler automatically splits its output into five memory sections: *code*, *strings*, *initialized data*, *constant data*, and *uninitialized data*. At link time you can locate each section into ROM or RAM as needed.

2. It's Standard

CrossCode C tracks the emerging ANSI C standard, and will conform to the standard when it becomes available. That means *your* code will always be standard, too.

CrossCode C already supports many ANSI C enhancements including enumerations, type *void*, structure assignments, and separate name pools for members of different structures. Also, structures may be passed as arguments and returned from functions in a fully reentrant fashion.

Function prototyping is available, so you can have the compiler automatically check the types of function arguments. Wherever possible, the compiler automatically casts arguments into the proper type, so you'll never spend your valuable time debugging improperly passed parameters.

No corners have been cut. CrossCode C is a fully reentrant, modern implementation of the C language.

3. Integrates C and Assembly Language

With CrossCode C, you can write your code in any combination of C and assembly language. The documentation tells you precisely how to pass parameters back and forth between assembly language routines and C functions.

Assembly language code can also be placed directly in-line with C code. Here, a "TRAP #1" occurs if *i* and *j* are equal:

```
if( i == j ) asm(" TRAP #1");
```

4. Generates Readable Assembly Language

While debugging, you'll want to know which C statements generated what assembly language code. So the CrossCode C compiler generates assembly language code *with your C language source code embedded as comments*. Statement by statement, you can see exactly how the compiler has translated your program.

When you declare a register or stack variable, the assembly listing shows you where the compiler has allocated it. In the example below, variables *g* and *h* have been assigned to registers D7 and D6, and *i* has been assigned a stack location of "-0x4(A6)":

```

C source
; register char g,h;
; D7 <- g
; D6 <- h
; int i;
; -0x4(A6) <- i

```

register allocation

stack allocation

Each C statement is followed by the code it emits:

```

C source
; i = ++b;
generated code {
ADDQ.B #0x1,D6
MOVE.B D6,D0
EXT.W D0
EXT.L D0
MOVE.L D0,-0x4(A6)
}

```

5. Generates Compact Code

CrossCode C always uses minimum required precision when evaluating expressions, so there's no unnecessary casting to *int*. Example:

```

; int i; char c1,c2;
; -0x4(A6) <- i
; -0x5(A6) <- c1
; -0x6(A6) <- c2
; c1 = i + c2;
byte instructions {
MOVE.B -0x1(A6),D0
ADD.B -0x6(A6),D0
MOVE.B D0,-0x5(A6)
}

```

Other standard optimizations include "folding" constants at compilation time, converting multiplications to shifts wherever possible, and eliminating superfluous branches.

6. Lets YOU Optimize Your Application

You can optimize the compiler's output for your application because you control the sizes of C types. For maximum integer arithmetic speed you can select two byte *ints*, or you can get maximum versatility by using four byte *ints*. You may also select 32 or 64 bit sizes for each floating point type.

If your load fits in 64K, you can get much tighter, faster code by selecting the *small-model* option. In small-model, all function calls are performed with "short" instructions and all data is accessed with 16 bit addresses.

7. Optimizes Via Heavy Register Use

CrossCode C reserves plenty of registers to hold your "register" variables. And there's a compiler option to automatically declare all stack variables as "register", so you

can instantly optimize programs that were written without registers in mind.

Within each C function, you can use six data registers (D2-D7) to store integral data types such as *char*, *int*, and *long*. Four address registers (A2-A5) are available for pointers. If you're using the 68881, you can use six of its registers (F2-F7) to hold your *floats* and *doubles*.

And there's a special feature for real-time applications: you can reserve any of the above registers for use as *global* register variables. Such variables retain their values across function calls, and may be used to hold time-critical data.

8. Supports the 68020

If you're using the 68020, CrossCode C will use its extra instructions and addressing modes. In this 68000 example, an array element is retrieved using three instructions, and a "long" multiplication must be performed via a library routine:

```

; register long i; long buf[8];
; D7 <- i
; -0x20(A6) <- buf
; i *= buf[i];
MOVE.L D7,D0
MOVE.L D7,D2
LSL.L #0x2,D2
MOVE.L -0x20(A6,D2.L),D1
JSR _lmul
MOVE.L D0,D7

```

retrieve array element

perform multiplication

With the "68020" option set, the following code is generated:

```

; i *= buf[i];
MULU.L -0x20(A6,D7.L*4),D7

```

9. Provides 68881 Floating Point Support

CrossCode C is fully compatible with the 68881 coprocessor because all floating point constants are stored in IEEE STD-754 format. If you're using the 68881, the compiler performs all floating point operations through the coprocessor, and *float* and *double* register variables are stored in 68881 registers.

If you're not using the 68881, the CrossCode C library provides floating point routines for all C language floating point operators (i.e. +, -, *, /). However, you must supply your own subroutines for any higher order floating point functions (e.g. *sin*, *sqrt*) required by your application.

10. Comes With C Library Source

An extensive C library containing over 47 C functions is provided in source form. You may add, modify, or delete library functions. Included are routines for string formatting (*sprintf/printf*), string copying and comparison (*strcpy*), memory allocation (*malloc*), and state recovery (*setjmp/longjmp*).

11. No Limitations

No matter how large your program is, CrossCode C will compile it. The compiler imposes no limits on the number of symbols in your program, the size of your input file, or

the size of a C language function. Symbol names may contain up to 64 significant characters, and large symbol tables overflow gracefully to disk.

12. Supports Position Independence

CrossCode C can generate position independent code and data. Small and large models can be selected for any target processor. Small models use 16 bit offsets for tighter code, while large models allow unlimited code and data sizes.

Code, ROM data, and RAM data can be selected separately for position independence. Code is pc-relative. RAM data is relative to an address register, and ROM data may be accessed either way.

It's Tailor-Made

A truly ROMable package must start with a ROMable C compiler. CrossCode C gives you five independent compiler output sections, user-selectable type sizes, and a simple, well documented assembly language interface. In short, it has everything you need to get your code into ROM.

SOFTWARE DEVELOPMENT SYSTEMS, INC
 4248 BELLE AIRE LANE
 OWNERS GROVE, ILLINOIS 60515 USA
 Phone: 1-800-448-7733 or 1-312-971-8170
 Fax: 1-312-971-8513

Don Williams,
 68 Micro Journal,
 5900 Casandra Smith Road,
 Rixson, TN 37343

Dear Don,

A lot of water has flowed under the bridge since I last wrote on XBASIC XPLANATIONS, so I figure it's time for another letter - prompted by a discussion with a reader in New Zealand. I'd like to discuss how floating-point numbers are represented internally, which (if readers are interested!) could lead me on to discuss floating-point math operations in general!!

Let's start at the very beginning by looking at a simple decimal number, let's say 23.75. As it stands this is in fixed-point notation, the decimal-point being in its correct position. We could re-write this as 2.375×10^1 or 0.2375×10^2 , where the exponent (to the base 10) tells us how many places the decimal-point should be shifted to the right to give us the true number.

The two examples above could alternatively be expressed as 2.375E1 or 0.2375E2, given that 'E' means 'to the base 10'. The portion to the left of the 'E' is called the 'mantissa', while that to the right is called the 'exponent'.

In XBASIC, numbers are represented in binary, with a binary mantissa such that all significant digits are to the right of the DP (decimal-point), the leading '0.' being discarded, as it's understood. So, let's begin by translating our 23.75 into straight binary, or rather HEX. How do we go about this? There are two steps - one to translate the integer portion, 23, and the other to translate the fractional part .75.

Here's how we do it. First we write the number 23, with a vertical column to the right, and then successively keep dividing by 16 till we reach 0, noting each remainder to the right of the column.

| | | |
|----|--|--------------------------------|
| 23 | | B (remainder of 11 = B in HEX) |
| 1 | | B |
| 0 | | 1 |

Then, reading from the bottom up, we get 1B, which is 27 in decimal. Now for the fractional part, which is the reverse of what we've just done, except that we first translate to binary. That is, we draw a vertical column with the decimal fraction to the right, and keep multiplying the portion to the right by 2 (till we reach 00 to the right) noting the overflow to the left of the column. So :

| | | |
|---|--|----|
| 1 | | 75 |
| 1 | | 50 |
| 1 | | 00 |

Reading from the top downwards, we get .11. The complete number is therefore 1B (that is, 1 1011) followed by our fractional 11, which gives us 11011.11, or, more correctly for floating-point, we have 11011.11×2^2 , or 11011.11E0, where 'E' now means to the base 2.

Before proceeding, let's look at our 11011.11 in a little more detail. As everyone knows, the value of each '1' in this notation multiplies by 2 as we progress leftwards from the decimal-point (or should I now call it a binary-point?), so that we have a '1', plus a '2', plus no '4', plus an '8', plus a '16', which adds up to our integer-part, 27. In like fashion, as we move right from the BP, we progressively divide the value by 2, so that we have .5 plus .25, to give us our .75.

Now let's slide our BP to the extreme left of our number (5 steps altogether) to give 0.1101111E5, the E5 telling us that the BP has to be moved to the right by 5 places to restore the number to its correct form.

Internally, BASIC drops '0.', and converts the binary number remaining to a HEX-format, by dividing it into blocks of 4 bits (commencing at the left) and padding it out to a full 7 bytes (note, bytes - not bits) with the exponent stuck on the end as the 8th byte.

Let's do that! We have 1101 1110 as our first byte, followed by a string of 00s, with a final 05 for the exponent, or DE 00 00 00 00 00 05. This is only the beginning, however, as we now have to translate this into a "suppressed-bit" format. That is, just as we dropped the '0.' because that would always be the situation at the extreme left of our number, we can also "suppress" the left-most '1', as this, too, will always be so. The first byte of our number will therefore become 0101 1110, or 5E instead of DE.

What's the point of doing that?, you ask. Well, now we can represent negative numbers, by using the vacated most-significant bit as a sign-bit, so that if our original number had been -23.75 it would now be represented as DE (plus all the trailing 00s, of course). We still haven't finished, but I've run out of space, so we'll continue next month. Keep in mind that all the above applies equally as well to our own 6809 RBASIC, and to our 68000 RBASIC, except that our 68K version (because of its greater precision) has an 8-byte mantissa and a 2-byte exponent.

Sincerely,



R. Jones
 President

Micronics Research Corp, 33383 Lynn Ave., Abbotsford,
 B.C., CANADA V2S 1E2



Contact: Bob Anundson, President
88open Consortium, Ltd.
503/ 682-5703

88open Spring Meeting in the European Computer Marketplace 88000 RISC Systems and Software Explored

Portland, Oregon, USA, March 13, 1989 -- 88open Consortium Ltd., announced its upcoming general meeting in Europe, to be held May 9 and 10. The Grand Hotel Saltsjöbaden, located near Stockholm, Sweden, is the venue for 88open's European general session. Registration and a welcome to Sweden hospitality function are scheduled for May 8, starting at 4:00. The schedule of events for May 9, includes an overview of 88open and the Software Initiative mission and goals. Plus update sessions on the Motorola 88000 RISC chip, updates on OSF and Unix International, plus product status presentations, and a reception sponsored by Motorola. The schedule for May 10, includes a product fair, Software Initiative break out sessions, 88open committee overviews, and tutorials on BCS/OCS, embedded systems and support chips.

The May venue in Sweden is the perfect occasion to learn about the 88K systems and software marketplace in Europe. It's worth noting that Unix is increasingly a standard operating system in Europe and some governments will soon standardize on Unix. Britain is a buoyant Unix market, as are Sweden, Norway, Denmark, and Finland.

With a number of European members, such as Edinburgh Portable Compilers (Scotland), Ericsson (Sweden), Integrated Micro Products (UK), Tadpole Technology (UK), and Dolphin (a subsidiary of Norsk Data, Norway), 88open is uniquely positioned to serve the needs of the 88K community in Europe, as well as provide valuable information to other members of the Consortium.

The 88open Consortium Ltd., is a not-for-profit organization formed to develop and promote the success of Motorola's 88000 RISC microprocessor architecture. The Consortium, based in Wilsonville, Oregon, has more than 50 members worldwide. 88open is pioneering a unique method of doing business in the computer industry based on the profitability of 88000-based applications across a variety of hardware platforms including fault tolerant systems, computer servers, workstations and personal computers.

88open members and potential members are invited to attend this important general meeting in Stockholm to learn about the European market, 88open, and the Software Initiative. Reservations must be received by the Grand Hotel Saltsjöbaden by April 2. Contact 88open for complete reservation information. Telephone +1 503 682 5703. Fax +1 503 682 5836. Mailing address: 88open Consortium Ltd., 8560 SW Salish Lane, Suite 500, Wilsonville, Oregon 97070 USA.

88open Consortium Announces Software Initiative

New Business Methods to Create Shrink Wrapped RISC Applications for the Unix Market

BURLINGAME, Calif. — Feb. 21, 1989 — The 88open Consortium Ltd. today announced the formation of the Software Initiative, an organization designed to accelerate compatible software development for the Motorola 88000 RISC architecture. In a related announcement, more than 26 independent software vendors (ISVs) stated that they will deliver new applications for the 88000 architecture in 1989.

The Software Initiative is composed of hardware vendors including Data General, Motorola Inc., NCR and Sanyo/Icon International.

The Software Initiative was created by hardware systems vendors who wanted to accelerate and support the process of software development. The Initiative members have focused on creating an environment that allows 88000 RISC-based application software to operate transparently across numerous systems, similar to PC-clone "shrink wrapped" software. To achieve this, the Software Initiative will provide ISVs with equipment and technical assistance.

"We see tremendous acceptance and support for the 88000 by the software community," said Bob Anundson, executive director of the 88open Consortium. "This is due not only to the large potential market generated by the 88000, but also to the standards that make it straightforward to access all the systems using the 88000, and not just a few manufacturers."

Two significant purposes of 88open provided the genesis of the Software Initiative:

- 1) Fostering the rapid porting of software to the standard computer environment.
- 2) Designing, implementing and promoting a compliance testing and labeling program that will ensure the end user that the products, both hardware and software, which are marked as compliant are interoperable.

The Software Initiative will monitor and administer the certification process for compliant products. 88open will approve ISV self-testing procedures for a software port, after which the ISV will perform its own compliance tests before submitting the product and test results to 88open. The Consortium will review (and audit) compliance and issue a trademarked seal to compliant software. A similar program for hardware will require execution of a custom test suite as well as compliant applications software to be eligible for certification.

"We are extending standards and the standards development process beyond the norm established by the industry," said Roger Cady, director of the Software Initiative. "Our organization is providing ISVs with more than just standards on a piece of paper. We have committed people, resources and funding necessary to bring full implementation of 88000-based hardware and software to all computing markets."

88open Consortium Ltd.

8560 SW Salish Lane • Suite 500 • Wilsonville, OR 97070 • USA • tel (503) 682-5703 • fax (503) 682-5836



Andrew Hettlinger
508-898-4060

DATA GENERAL ANNOUNCES DG/UX 4.1, INDUSTRY'S FIRST UNIX OPERATING SYSTEM FOR FAMILY OF RISC-BASED SYSTEMS

SAN FRANCISCO, Feb. 28 -- Data General today announced DG/UXTM Revision 4.1, the first UNIX operating system capable of running on a full family of RISC machines from single-user workstations to multiprocessor-based systems. Data General also announced its intent to separately license DG/UX 4.1 on other vendors' systems.

DG/UX 4.1 is an enhanced version of the DG/UX operating system, that embraces industry standards and is designed to take full advantage of the Motorola 88000 RISC architecture. "With this announcement, Data General delivers on its commitment to port the industry-leading DG/UX 4.0 operating system from MV/Family systems to the new family of 88000-based products," said Herb Osher, division director of product marketing.

DG/UX 4.1 is a commercial-grade implementation of UNIX, compatible with both UNIX System V.3 Release 1 from AT&T and Berkeley Software Distribution Revision 4.2. It incorporates the major features found in UNIX operating systems with all of today's available UNIX standards.

DG/UX 4.1 complies with the following industry, government or de facto standards:

- SVID2 (System V Interface Definition Issue 2)
- SVVS (System V Verification Suite)
- POSIX (IEEE Portable Operating System Interface Specification P1003.1)
- 88open Binary and Object Compatibility Standards
- X Windows Version 11 release 3.0
- ONCTM/NFS 4.0

DG/UX 4.1's advanced features support workstations, servers and large multiprocessor systems, in technical and commercial environments, with unmatched performance for computational and interactive applications. These features, which are transparently available to applications, include: fully-symmetric multiprocessor support for greater processing power, a re-engineered kernel providing greater portability and maintainability of system code, a robust file system for greater data security and recoverability, a flexible file system that permits the management of files for growth, performance, and convenient backup, an intelligent scheduler for greater throughput and better response times than provided by ordinary UNIX schedulers and diskless/tapeless workstation support for lower cost and centralized control of data.

For compatibility with existing applications, DG/UX implements 108 of the 111 portable application-callable BSD system calls and the most useful BSD libraries and commands, an ANSI C Compiler optimized for the 88000 architectures, eight-bit international character set support and an extensive range of easy-to-use system administration facilities.

DG/UX 4.1 supports a wide range of layered software products including TCP/IP communications software, NFS 4.0 and X-Windows Version 11 release 3. DG/UX 4.1 can execute BCS-compliant applications developed for 88000-based systems and has the ability to generate and execute SVID-, POSIX- and BSD-compliant applications in BCS format.

888

```

DOFLS6 CLR    GOTOFL,PCR    CLEAR GOTO FLAG
        LDA    SAVECH,PCR    RESTORE ECHO STATUS
        STA    ECHOFL,PCR
        RTS

*
DOTRUE  TST    NOTFL,PCR    TRUE; TEST 'NOT' FLAG
        BNE    DOFLS1
DOTRU1  BSR    CHKPAR        DEFINITELY TRUE; CHECK PARAMETER
        CMPB   $PARTN        IF 'THEN'
        BNE    DOTRU2
        INC    IFLEV,PCR    THEN INCREMENT 'IF' NESTING LEVEL
        RTS

*
DOTRU2  CMPB   $PARGTO        ELSE IF 'GOTO', GOTO LABEL
        LREQ   DOGOTO
        LBRA   CMDPER        ELSE HAVE ERROR

*
* PROCESS 'NOTE' COMMAND
*
DONOTE  TST    ECHOFL,PCR    IF ECHO IS OFF
        BNE    DONOT2
DONOT1  LDA    $504
        STA    {TRMADR,PCR}  CHANGE LINE TERMINATOR TO EOT
        LDX    LINPTR        THEN OUTPUT REST OF LINE TO THE TERMINAL
        JSR    PSTRNG
DONOT2  RTS

*
* PROCESS 'ON' COMMAND
*
DOON    BSR    CHKPAR        CHECK PARAMETER
        CMPB   $PARERR        IF NOT 'ERROR' THEN HAVE ERROR
        LBNZ   CMDPER
        BSR    CHKPAR        CHECK PARAMETER
        CMPB   $PARCON        IF 'CONTINUE'
        BNE    DOON1
        CLR    ERRFLG,PCR    THEN CLEAR ERROR FLAG
        RTS

*
DOON1   CMPB   $PARGTO        ELSE IF NOT 'GOTO'
        LBNZ   CMDPER        THEN HAVE ERROR
        LEAX   ERRLAB,PCR    ELSE GET ERROR LABEL
        CLR    0,X
        LDB    #8
DOON2   JSR    NXTCH
        STA    ,X+
        CMPA   $SPACE
        BLS    DOON3
        DECB
        BNE    DOON2
DOON3   TST    ERRLAB,PCR    IF NO LABEL THEN HAVE ERROR
        LBRQ   CMDPER

```

```

        LDA    #1            SET ERROR FLAG
        STA    ERRFLG,PCR
        RTS

*
* PROCESS 'PAUSE' COMMAND
*
DOPAUS  BSR    DONOTE        OUTPUT REST OF LINE TO THE TERMINAL
        LEAX   CONTM,PCR    OUTPUT CONTINUE MESSAGE
        JSR    PSTRNG
        JSR    INCH2        WAIT FOR CHAR FROM KEYBOARD
        RTS

*
* PROCESS 'REM' COMMAND
*
DOREM   RTS                IGNORE REST OF LINE

*
* PROCESS 'SHIFT' COMMAND
*
DOSHFT  LEAY   PARPTR+2,PCR   POINT TO FIRST COMMAND LINE PARAMETER (%1)
        LDB    #NPARAM-1
DOSHI   LDX    2,Y            SHIFT PARAMETER
        STX    ,Y++
        DECB
        BNE    DOSHI
        RTS

*
*****
* COMMAND AND PARAMETER LOOKUP ROUTINES
*
*****
* CHECK FOR PARAMETER
* RETURN ACCB = PARAMETER NUMBER, OR ZERO IF NOT FOUND
*
CHKPAR  LEAX   PTABLE,PCR
        BSR    CHKTBL
        TFR    X,D
        RTS

*
* CHECK FOR COMMAND
* RETURN IX = ROUTINE ADDRESS, Z = 0 IF FOUND; Z = 1 IF NOT
*
CHKCMD  LEAX   CTABLE,PCR
        PSHS   X            SAVE ADDRESS OF CTABLE
        BSR    CHKTBL        LOOK FOR COMMAND
        PUIS   D
        BEQ    CHKCM1
        LEAX   D,X          ADD IN ADDRESS OF CTABLE

```

```

CHKCMI RTS
*
*
* CHECK TABLE FOR ENTRY
* ITEM TO CHECK IN LINPTR -> LINBUF, IX -> TABLE TO CHECK AGAINST
* RETURN ENTRY VALUE IN IX, Z = 0 IF FOUND; IX = 0, Z = 1 IF NOT
*
CHKTBL LDY LINPTR SAVE LINPTR
CHKTBL PSHS Y
BSR COMPARE TWO ITEMS
BNE CHKFND
LEAX 2,X IF DIFFERENT, MOVE TO NEXT TABLE ENTRY
PULS Y RESTORE LINPTR
STY LINPTR
TST 0,X IF FIRST CHAR OF TABLE ENTRY <> 0
BNE CHKTBL1 THEN CONTINUE
*
LDX #0 NOT FOUND; RETURN IX = 0, Z = 1
RTS
*
CHKFND LEAS 2,S FOUND; RESTORE STACK
LDX ,X RETURN IX = TABLE ENTRY VALUE AND Z = 0
RTS
*
*
* COMPARE TWO ITEMS TERMINATED BY <= SPACE, POINTED TO BY IX
* RETURN IX POINTING TO BYTE PAST TERMINATOR
* RETURN Z = 0 IF SAME, Z = 1 OTHERWISE
*
COMPARE JSR NXTCH GET NEXT CHAR FROM LINBUF
CMPA #SPACE IF <= SPACE, END LOOP
BLS CMPAR4
BSR TOUPPR CONVERT CHAR TO UPPER CASE
PSHS A
LDA ,X+ GET NEXT CHAR FROM STRING
BSR TOUPPR CONVERT TO UPPER CASE
CMPA ,S+ COMPARE TWO CHARS
BEQ COMPARE
CMPAR2 LDA ,X+ NOT SAME; MOVE IX TO BYTE PAST TERMINATOR
CMPA #SPACE
BHI CMPAR2
CMPAR3 CIRA SET Z = 1
RTS
*
CMPAR4 LDA ,X+ CHECK IF IX -> TERMINATOR
CMPA #SPACE
BHI CMPAR2 IF NOT, NO MATCH
LDA #1 SET Z = 0
RTS
*
*
* CONVERT CHAR TO UPPER CASE
*

```

```

TOUPPR CMPA #'a
BIO TOUPP1
CMPA #'z
BHI TOUPP1
SUBA #$20
TOUPP1 RTS
*
*****
*
* COMMAND FILE LINE INPUT AND PARAMETER EXPANSION ROUTINES
*
*****
*
* READ NEXT LINE FROM THE COMMAND FILE
*
READLN TST ECHOFL,PCR IF ECHO IS ON
BEQ READL0
JSR PCRLF THEN OUTPUT CRLF AND PROMPT CHAR
LDA #'+'
JSR OUTCH2
READL0 CLR INPARM,PCR CLEAR FLAGS
CLR INLINE,PCR
CLR INTEXT,PCR
LDY #LINBUF START OF LINE BUFFER
STY LINPTR
READL1 BSR READCH GET CHAR
STA ,Y+
CMPA #CR
BEQ READL3
CMPY #LINBUF+127 IF END OF BUFFER
BNE READL2
LEAY -1,Y THEN DECREMENT BUFFER POINTER WAITING FOR
CR
READL2 BRA READL1
*
READL3 LDD #$0D0D ADD SECOND TERMINATOR
STD ,Y-
STY TRMADR,PCR SAVE FIRST TERMINATOR ADDRESS
LDA #$FF INDICATE NEXT CHAR = START OF LINE
STA STLINE,PCR
RTS
*
*
* READ NEXT CHAR FROM COMMAND FILE, PARAMETER BUFFER OR TERMINAL
*
READCH PSHS X,Y,B
TST INPARM,PCR CHECK EXPANDING PARAMETER FLAG
BEQ READC2
*
* READ FROM PARAMETER
*
LOX CURPTR,PCR GET NEXT CHAR FROM PARAM BUFFER
READC0 LDA ,X+

```

```

FILE      BEQ      READC1      IF ZERO, HAVE END OF PARAM; GET CHAR FROM
          STX      CURPTR,PCR
          BRA      READC8      ELSE CONTINUE
*
* READ FROM COMMAND FILE
*
READC1    CLR      INPARM,PCR    RESET EXPANDING PARAM FLAG
READC2    TST      INLINE,PCR    IF TERMINAL LINE FLAG SET
          BNE      READC7      THEN READ FROM TERMINAL
          BSR      CMDCHR      GET NEXT CHAR FROM COMMAND FILE
          CMPA     #'%'      IF PARAMETER INDICATOR
          BNE      READC8
          BSR      CMDCHR      THEN GET NEXT CHAR
          CMPA     #'0'      CHECK FOR VALID PARAMETER
          BLO      READC3
          CMPA     #'9'
          BHI      READC3
          SUBA     #'0'      CONVERT CHAR TO PARAMETER NUMBER
          LEAX     PARPTR,PCR    GET PARAMETER POINTER
          ASIA
          LDX      A,X
          INC      INPARM,PCR    INDICATE EXPANDING PARAMETER
          BRA      READC0      GO AND GET CHAR FROM PARAMETER
*
READC3    TST      GOTOFL,PCR    IF GOTO FLAG NOT SET
          BNE      READC9
          CMPA     #'C'      THEN IF CHAR = 'C' OR 'c'
          BEQ      READC4
          CMPA     #'c'
          BNE      READC5
          LDA      #'?'      THEN OUTPUT PROMPT (?)
          JSR      PUTCHR
READCC    JSR      INCH2      READ CHAR FROM THE TERMINAL
          BRA      READC9
*
READC5    CMPA     #'L'      ELSE IF CHAR = 'L' OR 'l'
          BEQ      READC6
          CMPA     #'l'
          BNE      READC4
          INC      INLINE,PCR    THEN SET TERMINAL LINE FLAG
          BRA      READC4      AND GO AND READ CHAR
*
READC4    CMPA     #'T'      ELSE IF CHAR = 'T' OR CHAR = 't'
          BEQ      READC8
          CMPA     #'t'
          BNE      READC8
          INC      INTEXT,PCR    THEN SET TERMINAL TEXT FLAG
          INC      INLINE,PCR    AND TERMINAL LINE FLAG
          BRA      READCC      READ CHAR
*
READC7    JSR      INCH2      READING LINE FROM TERMINAL; GET CHAR
          TST      INTEXT,PCR

```

```

          BNE      READC9
          CMPA     #CR
          BNE      READC9
          CLR      INLINE,PCR    IF CHAR = CR
          BRA      READC9      THEN CLEAR TERMINAL LINE FLAG
*
READC8    TST      ECHOFL,PCR    IF ECHO FLAG SET THEN ECHO CHAR TO USER
          BEQ      READC9
          PSHS     A
          JSR      PUTCHR
          PULS     A
          PULS     X,Y,B
          RTS
*
READER    LEAX     PREFM,PCR      REPORT PARAMETER ERROR AND EXIT
          BRA      PERROR
*
* GET NEXT CHAR FROM COMMAND FILE
*
CMDCHR    LEAX     DOFCB,PCR      READ NEXT CHAR FROM COMMAND FILE
          JSR      FMS
          BEQ      CMDCH1
          IDB      F.ERR,X
          CMPB     #E.EOF
          BNE      DSKERR      THEN REPORT DISK ERROR AND EXIT
          TST      GOTOFL,PCR    ELSE IF GOTO FLAG SET
          BEQ      CMDCH0
          LEAX     NOLABM,PCR    THEN REPORT MISSING LABEL MESSAGE AND EXIT
          BRA      PERROR
*
CMDCH0    TST      IFLEVL,PCR      ELSE IF 'IF' NESTING LEVEL = 0
          BEQ      EXIT            THEN EXIT
          LEAX     INIFM,PCR      ELSE REPORT IF NESTING ERROR MESSAGE AND
          BRA      PERROR
*
CMDCH1    CMPA     #$02            IF BINARY START OF FILE MARKER
          BNE      CMDCH2
          LEAX     FILERM,PCR      THEN REPORT FILE TYPE ERROR AND EXIT
          BRA      PERROR
*
CMDCH2    TST      STLINE,PCR      IF START OF LINE FLAG SET
          BEQ      CMDCH3
          CMPA     #SPACE
          BEQ      CMDCHR      THEN IF CHAR = SPACE
          CLR      STLINE,PCR      THEN GET NEXT CHAR
          ELSE CLEAR START OF LINE FLAG
          RTS
*
*****
*
* ABORT AND ERROR ROUTINES
*

```



```

*****
*
* TEST FOR ABORT FROM THE USER
*
TSTAB1 JSR     STAT      IF KEY PRESSED
        BEQ     TSTAB1
        JSR     INCH2    THEN GET CHAR
        CHPA    #BRKCHR  IF CHAR <> BREAK CHAR
        BEQ     TSTAB2
TSTAB1 RTS          THEN RETURN
*
TSTAB2 LEAX     BREAKM,PCR ELSE PRINT BREAK MESSAGE AND EXIT
        BRA     PERROR
*
*
* COMMAND FILE PARAMETER ERROR
*
CMDPER LEAX     CMDPEM,PCR PRINT COMMAND PARAMETER ERROR AND EXIT
        BRA     PERROR
*
*
* PRINT DISK ERROR MESSAGE AND EXIT
*
DSKERR LEAX     DOFCB,PCR  REPORT DISK ERROR
        JSR     RPTERR
        BRA     PERR1
*
*
* PRINT ERROR MESSAGE AND EXIT
*
PERROR JSR     PSTNG      PRINT ERROR MESSAGE
PERR1  LEAX     ABORTM,PCR
        JSR     PSTNG
        JSR     PCRLF
*
*
* EXIT FROM 'DO'
*
EXIT   LDS     SAVESP,PCR  RESTORE STACK POINTER
        JSR     FMSCLS    ENSURE ALL FILES ARE CLOSED
        LDA     SVPAUS,PCR RESTORE TTY PAUSE FLAG
        STA     TTYP
        LDA     SVSPIO,PCR RESTORE SPECIAL IO FLAG
        STA     SPECIO
        LDD     MEMEND    IF MEMEND OK
        CPD     NEWEND,PCR
        BNE     EXIT1

```

```

LDD     SVMEND,PCR  THEN RESTORE MEMEND
STD     MEMEND
BRA     EXIT2
*
EXIT1  LEAX     MENDM,PCR  ELSE OUTPUT ERROR MESSAGE
        JSR     PSTNG
        JSR     PCRLF
EXIT2  CLR     CURLIN    CLEAR CURRENT LINE NUMBER ON PAGE
        CLR     CMDFLG   CLEAR COMMAND FLAG
        JMP     MARKS    RETURN TO FLEX
*
ENDDO  BQU     *
*
END     DO

```

```

+++

```

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Classifieds As Submitted - No Guarantees

Surplus Unused Motorola VME Modules & Electronic Solutions Enclosures for Sale at Discount

| | | |
|-----------|--|-------|
| MVME133 | CPU Module-68020, 1MB DRAM, 68881 FPP, | \$675 |
| | 3 serial Ports, EPROM Sockets, VMEbus Interface | |
| MVME225-1 | 1MB DRAM Module, A32/D32 VMEbus Interface | \$380 |
| MVME320A | Winchester / Floppy Controller | \$490 |
| MVME332 | 8 Channel intelligent Serial Communications Module | \$675 |
| Series 7 | Electronic Solutions 7 Slot Desktop Enclosure, P1/P2 | \$695 |
| | Backplane, 325W PS, Space for Winchester/Floppy/Tape | |

Respond to: John Gannon, RPG, P.O. Box C12399, Ste 162, Scottsdale, Arizona 85267 Phone (602) 951-3373

...

S+ Memory Cards, CPU Cards, Hard Disks w/Controller Cards, I/O Cards, Cabinets, Power Supplies.

S/09 CPU Cards, Memory, I/O Cards, Controller Cards, Cabinets, Power Supplies

3-Dual 8" drive enclosure with power supply. New in box. \$125 each.

5-Siemens 8" Disk Drive, \$100 each.

Tom (615) 842-4600 M-F 9AM to 5PM EST

...

HELIX 6809 SYSTEM, 6809 CPU card, 64K memory card, S-64 motherboard w/ 2 parallel, 2 serial.

Offer. (301) 977-5633

...

MUSTANG08, HD controller, OS968K, C, UMACS, US\$900. Phone me around 1900 UT or at 0600 UT NEW ZEALAND 04-847-596

FLEX™/SK-DOS™/MS-DOS™ Transfer Utilities

For 68020 and CoCo* OS-9 Systems

Now READS

WRITE -DIR - DUMP - EXPLORE

FLEX, SK-DOS & MS-DOS Disk

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: \$69.95 68020 Version \$99.95

S.E. Media

PO Box 849

5900 Cassandra Smith Rd.

Hixson, TN 37343

(615) 842-7990

FAX (615) 842-4600



SPECIAL - ATARI™ & OS-9™

NOW! - If you have either the Atari 520 or 1040 - you can take advantage of the "bargain of a lifetime" OS-9 68K and BASIC all for the low, low price of:

\$150.00

Call or Write

S.E. Media

5900 Cassandra Smith Rd.

Hixson, TN 37343

615 842-4601

FAX (615) 842-7990

SK*DOS®/68K

Read the fine print to see what's in SK*D S/68K:

☐ Full DOS documentation plus on-line help ☐ Multiple directories
☐ User-installable device drivers ☐ Install up to 8 different I/O devices ☐ Keyboard type-ahead ☐ Print-screen ☐ Virtual (RAM) disk ☐ Disk cache ☐ Up to 10 drives ☐ 5¼" or 3½" floppy drives ☐ Hard drives to 64 megabytes each ☐ I/O redirection to drives or I/O ☐ Time/date stamping of files ☐ File or disk write protect (even hard disk) ☐ Batch files ☐ Support for 68000, 68010, 68020 ☐ Monochrome or color video board support ☐ Read and write MS-DOS disk files ☐ 6809 Emulator ☐ Powerful utilities such as copy, by-date, undelete, show differences between files, prompted delete, text file browse, and more - all included ☐ Simple Basic included ☐ Fast assembler included ☐ Line editor included ☐ User support via newsletter and BBS ☐ Available software: C compiler, full Basic, screen editors, disassemblers, cross-assemblers, spelling checker, text formatter, music editor, hard disk manager, ROM-based debugger, modem communications programs, etc. More compilers coming. ☐ (Some features may not be implemented in all hardware manufacturers' implementations.)

Individual copies of SK*DOS/68K are \$140; less in quantity or when bundled with hardware. Send for our 6809 / 68K hardware and software catalog. Also available as part of our hardware/software educational course.



Software Systems Corp.
 P. O. Box 209J
 Mt. Kisco, NY 10549
 (914) 241-0287
 BBS (914) 241-3307 ♦ Fax (914) 241-8607

THE QUICK ED OS-9 SCREEN EDITOR

Word Processor and Text Formatting System

- ★ Edits up to nine files simultaneously
- ★ Microjustifies mixed proportional text
- ★ Uses all the fonts on your printer *Vi&E*™
- ★ Uses cursor and function keys on any terminal
- ★ Definable high-level formatting commands
- ★ User-specific keyboard mapping and macros
- ★ Highly configurable on multi-user systems
- ★ Automatic hyphenation and table of contents
- ★ Imports graphics and listings into documents
- ★ 450 pages of comprehensive documentation

At last ! Use your laser printer !

The Quick Ed screen editor and formatter: \$275.00

Call for information on OS-9 consultancy services, disk caching, graphics, lint, Modula-2 and file managers.
 'OS-9' is a Trademark of Microware Inc.



2407 Lime Kiln Lane, Louisville, KY 40222 U.S.A.
 Telephone: (502) 425 9580 Fax: (502) 425 3044

SOFTWARE

68000 C CROSS-COMPILER

\$100 - SKDOS,MSDOS,UNIX,XENIX (OBJECT ONLY)

Assembles K&R C language, generates 68000 assembly code. Includes 68010 cross assembler, libraries provided for SKDOS, but may be modified.

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES

EACH \$50-FLEX,OS9,UNIX,FLEX,MSDOS,UNIX,SKDOS,XENIX 1/8100 ALL/\$200

Specify: 1801, 6502, 6801/11, 6804, 6805, 6809, 28, 280, 8048, 8051, 8085, 68010, 32100. Modular cross-assemblers in C, with limited utilities. Sources for additional \$50 each, \$100 for 3, \$300 for all.

CMODEM TELECOMMUNICATIONS PROGRAM

\$100-MSDOS,SKDOS,UNIX,FLEX,OS9,XENIX,UNIX,FLEX OBJECT-ONLY: EACH \$50

Micro-driver with terminal mode, file transfer, MODEM7, XON-XOFF, etc.

SUPER SLEUTH DISASSEMBLERS

EACH \$99-FLEX \$101-OS9 \$100-UNIX/FLEX OBJECT-ONLY: EACH \$50-FLEX,OS9,COCO

Interactively generate source on disk with labels, includes xref, binary editing. Specify 6800,1,2,3,5,8,9/6502 version or 280/8080,5 version. COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not 280/8080,5) only. 68010 version \$100-FLEX,OS9,UNIX,FLEX,MSDOS,UNIX,SKDOS,XENIX.

DEBUGGING SIMULATORS FOR POPULAR 8-BIT CPUs

EACH \$75-FLEX \$100-OS9 \$80-UNIX/FLEX OBJECT-ONLY: EACH \$50-COCO FLEX, COCO OS9

Interactively simulate processors, includes disassembly, formatting, binary editing. Specify for 6800/1, (14)8085, 6502, 6809 OS9 only, 280 FLEX only.

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809 \$75-FLEX \$80-OS9 \$80-UNIX/FLEX
 6800/1 to 6809 & 6809 to pre-100 \$60-FLEX \$75-OS9 Only \$60-UNIX/FLEX

FULL-SCREEN KBASIC PROGRAMS with cursor control AVAILABLE FOR FLEX, UNIX/FLEX, AND MSDOS

| | |
|--------------------------------|----------------------------|
| Display Generator / Documenter | \$50 w/source, \$25 w/out |
| Mailing List System | \$100 w/source, \$50 w/out |
| Inventory with MRP | \$100 w/source, \$50 w/out |
| Tabular Rate Simulation | \$100 w/source, \$50 w/out |

DISK AND KBASIC UTILITY PROGRAM LIBRARY \$50-FLEX \$30-UNIX/FLEX/MSDOS

Edit disk vectors, sort directory, simulate master coding, do disk work, compile, sort or all BASIC programs, xref BASIC program, etc. non-FLEX versions include sort and resequencing only.

PROFESSIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING

We will customize any of the programs described in this advertisement or in our brochure for specialized company use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING

We will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of minicomputers, including IBM, Burroughs, Univac, Honeywell, most popular models of microcomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68010, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING

We offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants Inc.

1454 Little Lane
 Cary, Georgia 30507
 (404) 483-4570 • (404) 483-1717

Contact us about catalog, dealer, discounts, and services. Most programs in source give computer, OS, disk size. 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted. Add GA sales tax (if in GA) and 5% shipping. (UNIX/FLEX on Technical Systems Consultants; OS9 Microware COCO Tandy; MSDOS Microsoft; SKDOS Star Software)

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems*

*Hundreds Sold at
Suggested Retail:*

~~\$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compile their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the "Learn a New Language" syndrome - Write Your Program in BASIC, Debug It in BASIC and Then Compile It to a .CMD Binary File.

For a LIMITED time
save over 65%...
This sale will not be
repeated after it's
over! *

SALE SPECIAL:

\$69.95

SPECIAL

Thank-You-Sale

Only From:

CPI **S.E. Media™**
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000
FAX (615)842-7990

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

THE 6800-6809 BOOKS

OS - 9 User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibble

OS9 USER NOTES

Information for the BEGINNER to the PRO. Regular or CoCo OS9

Using OS9

HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building anew System Disk, OS9 Users Group, etc.

Programming Languages

Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include

No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK \$9.95

Typeset — w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder \$5.50

All Source Listings on Disk

1-8" SS, SD Disk \$14.95
2-5" SS, SD Disks \$24.95

FLEX USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the "Bible" for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the TEXT files included in the book and on diskette. All TEXT files in the book are on the disks.

| | |
|------------|--|
| LOGO.C1 | File load program to offset memory - ASM PIC |
| MOVES.C1 | Memory move program - ASM PIC |
| DUMP.C1 | Printer dump program - uses LOGO - ASM PIC |
| SUBTEST.C1 | Simulation of 6800 code to 6809, show differences - ASM |
| TERMEM.C2 | Modem input to disk (or other port input to disk) - ASM |
| M.C2 | Output a file to modem (or another port) - ASM |
| PRINT.C3 | Parallel (enhanced) printer driver - ASM |
| MODEM.C2 | TTL output to CRT and modem (or other port) - ASM |
| SCIPKG.C1 | Scientific math routines - PASCAL |
| U.C4 | Mini-monitor, disk resident, many useful functions - ASM |
| PRINT.C4 | Parallel printer driver, without PFLAG - ASM |
| SET.C5 | Set printer modes - ASM |
| SETBAS1.C5 | Set printer modes - A-BASIC |

Note: .C1, .C2, etc.=Chapter 1, Chapter 2, etc.

** Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: \$7.95 + \$2.50 S/H

With disk: 5" \$20.90 + \$2.50 S/H
With disk: 8" \$22.90 + \$2.50 S/H

Shipping & Handling \$3.50 per Book, \$2.50 per Disk set
Foreign Orders Add \$4.50 Surface Mail
or \$7.00 Air Mail

If paying by check - Please allow 4-6 weeks delivery
* All Currency in U.S. Dollars

Continually Updated in 68 Micro Journal Monthly
Computer Publishing Inc.
5900 Cassandra Smith Rd.

Hixson, TN 37343

(615) 842-4601

FAX (615) 842-7990

Telex 5106006630



FLEX is a trademark of Technical Systems Consultants
OS9 is a trademark of Microware and Motorola Telex 5106006630
68' Micro Journal is a trademark of Computer Publishing Inc.

68' Micro Journal

Reader Service Disks

- Disk-1 Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
- Disk-2 Diskedit w/ inst. & fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpaw, **Lifetime.
- Disk-3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
- Disk-4 Mailing Program, *Finddat, *Change, *Testdisk.
- Disk-5 *DISKFDX 1, *DISKFDX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
- Disk-6 **Purchase Order, Index (Disk file indx).
- Disk-7 Linking Loader, Rload, Harkness.
- Disk-8 Crest, Larpher (May 82).
- Disk-9 Datecopy, Diskfix9 (Aug 82).
- Disk-10 Home Accounting (July 82).
- Disk-11 Dissembler (June 84).
- Disk-12 Modem68 (May 84).
- Disk-13 *Initum68, Testum68, *Cleanup, *Diskalign, Help, Date.Txt.
- Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib.
- Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Conno).
- Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
- Disk-17 Match Utility, RATBAS, A Basic Preprocessor.
- Disk-18 Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMD.CODE, CMD.Txt (Sept. 85 Spray).
- Disk-19 Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
- Disk-20 UNDX Like Tools (July & Sept. 85 Taylor & Gilchrist), Dragon.C, Grep.C, L.S.C, FDUMP.C.
- Disk-21 Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
- Disk-22 Read CPM & Non-FLEX Disks. Fraser May 1984.
- Disk-23 ISAM, Indexed Sequential file Accessing Methods, Condon 11/85. Extensible Table Driven. Language Recognition Utility, Anderson 3/86.
- Disk-24 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
- Disk-25 KERMIT for FLEX derived from the UNDX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
- Disk-26 Compacta UniBoard review, code & diagram, Burlison March '86.
- Disk-27 ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
- Disk-28 CT-82 Emulator, bit mapped.
- Disk-29 **Star Trek
- Disk-30 Simple Winchester, Dec. '86 Green.
- Disk-31 *** Read/Write MS/PC-DOS (SK-DOS)
- Disk-32 Heir-UNIX Type upgrade - 68MJ 2/87
- Disk-33 Build the GT-4 Terminal - 68MJ 11/87 Condon.
- Disk-34 FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/89 Koipi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
 *** Denotes 68000 - 6809 no indicator.

8" disk \$19.50

5" disk \$16.95

Shipping & Handling - U.S.A. Add: - \$3.50
 Overseas add: \$4.50 Surface - \$7.00 Airmail

68' MICRO JOURNAL

5900 Cassandra Smith Rd.
 Hixson, TN 37343
 (615) 842-4600
 FAX (615) 842-7990
 Telex 510 600-6630



!!! Subscribe Now !!!

68' MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: ☐ Mastercard ☐ VISA

Card # _____ Exp. Date _____

For 1 Year _____ 2 Years _____ 3 Years _____

Enclosed: \$ _____

Name _____

Street _____

City _____ State _____ Zip _____

Country _____

My Computer is: _____

My Operating System is: _____

Subscription Rates

U.S.A.: 1 Year \$24.50, 2 Years \$42.50, 3 Years \$64.50

*Foreign Surface: Add \$12.00 per Year to USA Price.

*Foreign Airmail: Add \$48.00 per Year to USA Price.

*Canada & Mexico: Add \$9.50 per Year to USA Price.

*U.S. Currency Cash or Check Drawn on a USA Bank !

68' Micro Journal

5900 Cassandra Smith Rd.

POB 849

Hixson, TN 37343

Telephone 615 842-4600

FAX (615) 842-7990

Telex 510 600-6630



IF YOU NEED IT, WE'VE GOT IT! (OR WE'LL MAKE IT)

Yes, PERIPHERAL TECHNOLOGY still sells the FD-2 for SS-50 Bus Computers. And, if you don't need it, we sell other products from Single Board Computers to Systems which should fit your requirements. Customs Hardware Design is also available. Here's a small sample of what we offer:

FD-2 FLOPPY DISK CONTROLLER

- Controls up to four 5 1/4" Drives
- Runs in 1 or 2 MHz Systems
- Can be configured for either 4 or 16
- Addresses per I/O Slot (SS30 or SS30C)
- Uses WD2797 Controller Chip (compatible with 1771/179X Controller Chip)
- Hardware and Software compatible with SWTPC DC-4 controllers
- 6800/6809 Flex Drivers available
- SK'DOS Operating System or OS9/6809 Driver Package



OS9 is a Trademark of Microware & Motorola
SWTPC is a Trademark of Southwest Technical Systems

PT69-4 SINGLE BOARD COMPUTER

- 6809E Processor/1 MHz Clock
- Four RS232 Serial Ports using 6850'S
- Two 8-Bit parallel Ports using 6821 PIA
- Time-Of-Day Clock (MC146818)
- 59K of user RAM
- 2K or 4K of EPROM using 2716 or 2732
- Double Sided/Double Density Floppy Controller
- Can Read/Write Radio Shack OS/9 Diskettes
- Board Size 5.6" x 8.2"

MONTHLY SPECIAL
FD-2 Floppy Disk Controller: **\$129.00**
Regular Price: \$149.00

PERIPHERAL TECHNOLOGY

1710 Cumberland Point Dr. Suite B
Marietta, Georgia 30067
(404) 984-0742 Telex# 880584

VISA/ MASTERCARD/ CHECK/ C.O.D.

SYSTEMS

- Floppy or Winchester Versions available
- Systems use any 6809 Single Board Computer or 68008 Board
- Will be configured to meet your Requirements
- OS9 & SK'DOS Operating Systems
- Call or write for system configurations



Catalogues Available Upon Request.

DATA-COMP

SPECIAL

Heavy Duty Power Supplies



For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

Make: Boschert

Size: 10.5 x 5 x 2.5 inches

Including heavy mounting bracket and heatsink.

Rating: in 110/220 volts ac (snap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip

Load Reaction: Automatic short circuit recovery

SPECIAL: \$59.95 each
2 or more \$49.95 each

Add: \$7.50 each S/H

Make: Boschert

Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (snap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex

Load Reaction: Automatic short circuit recovery

SPECIAL: \$49.95 each
2 or more \$39.95 each

Add: \$7.50 S/H each

5900 Cessandre Smith Rd., Hixson, Tn. 37343

Telephone 615 842-4600

Telex 510 600-6630

Fax (615)842-7090



Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. \$250.⁰⁰ **Only \$79.⁹⁹**

STAR-DOS PLUS+

- Functions Same as FLEX
- Reads - writes FLEX Disks '34.⁹⁹
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

FLEX-CoCo Jr.
without TSC
Editor & Assembler
\$49.⁹⁹

PLUS

ALL VERSIONS OF FLEX & STAR-DOS INCLUDE

TSC Editor
Reg \$50.00
NOW \$35.00

- + Read-Write-Dir RS Disk
- + Run RS Basic from Both
- + More Free Utilities

- + External Terminal Program
- + Test Disk Program
- + Disk Examine & Repair Program
- + Memory Examine Program
- + Many Many More!!!

TSC Assembler
Reg \$50.00
NOW \$35.00

UPS Battery Backup

| | | |
|------|------------------|--------|
| 330 | VA UPS 4 Outlets | 309.00 |
| 520 | VA UPS 4 Outlets | 599.00 |
| 800 | VA UPS 6 Outlets | 859.00 |
| 1200 | VA UPS 6 Outlets | 999.00 |

Disks

| | |
|--------------------|----------------|
| 5" Box of ten (10) | |
| DS-DD | 5.99 40 Track |
| DS-DD | 14.95 80 Track |
| 3.5 DS-DD | \$15.95 |

Printers

| | |
|---------------|--------------------|
| EPSON | |
| LQ-500 24 Pin | 350. ⁰⁰ |
| LX-800 9 Pin | 239. ⁰⁰ |

Switch Boxes

| | |
|--------------------------------------|-------|
| Serial/Parallel Converter | 69.00 |
| Parallel Serial Converter | 69.00 |
| Serial 2 Position Switch Box | 27.00 |
| Parallel 2 Position Switch Box | 27.00 |
| Serial 4 Position Switch Box | 35.00 |
| Parallel 4 Position Switch Box | 33.00 |
| Serial 4 Position Crossover Switch | 44.00 |
| Parallel 4 Position Crossover Switch | 49.00 |
| Serial 2 Position 9 Pin Switch Box | 44.00 |

Gender Changer

| | |
|------------------|------|
| SERIAL | |
| Male to Male | 8.95 |
| Female to Female | 8.95 |
| Centronics | |
| Male to Male | 9.95 |
| Female to Female | 9.95 |

DATA-COMP
5900 CASSANDRA SMITH RD.
HIXSON, TN. 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN \$3.50



(615) 842-4600
FOR ORDERING
TELEX 5106006630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the \$500.00 OS-9 C compiler! This offer won't last forever!

**NOT 128K, NOT 512K
FULL 768K No Wait RAM**

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-88K™ and/or Peter Stark's SKDOS™. SKDOS is a single user, single tasking system that takes up where "FLEX" left off. SKDOS is actually a 68XXX FLEX type system (Not a TSC product.)

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed whiz on disk I/O. Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks!


System includes OS-9 68K or SKDOS - Your Choice Specifications:

| | | |
|-----------|--------------------|-------------------------|
| CPU | MC68008 | 12 Mhz |
| RAM | 768K | 256K Chips |
| | No Wait States | |
| PORTS | 4 - RS232 | MC88EB1 QUART |
| | 2 - 8 bit Parallel | MC8821 PIA |
| CLOCK | MC48T02 | Real Time Clock Bat. BU |
| EPROM | 16K, 32K or 64K | Selectable |
| FLOPPY | WD1772 | 5 1/4 Drives |
| HARD DISK | Interface Port | WD1002 Board |

**Now more serial ports - faster CPU
Battery B/U - and \$850.00 OS-9 Profes-
sional with C compiler included!**

***\$400.00**

See Mustang-02 Ad - page 5
for trade-in details



Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

LOOK

| | | |
|-----------------------------|-----------------------|----------|
| Seconds | 32 bit | Register |
| | Integer | Long |
| Other 68008 8 Mhz | OS-9 68K...18.0...9.0 | |
| MUSTANG-08 10 Mhz | OS-9 68K...9.8...6.3 | |
| Main() | | |
| { | | |
| C Benchmark Loop | | |
| { | | |
| /* Init I; */ | | |
| register long I; | | |
| for (I=0; I < 999999; ++I); | | |
| } | | |

**Now even faster!
with 12 Mhz CPU**

| C Compile times: OS-9 68K | Hard Disk |
|----------------------------|----------------|
| MUSTANG-08 8 Mhz CPU | 0 min - 32 sec |
| Other popular 68008 system | 1 min - 05 sec |
| MUSTANG-020 | 0 min - 21 sec |


**25 Megabyte
Hard Disk System**
\$2,398.90

Complete with PROFESSIONAL OS-9
includes the \$500.00 C compiler, PC
style cabinet, heavy duty power supply,
5" DDDS 80 track floppy, 25 MegByte
Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC

MUSTANG-08 is a trademark of CPI

Data-Comp Division



A Decade of Quality Service®
Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, TN 37343

* Those with SW/PC hi-density FLEX 5" - Call for special info.